Towards Dense Visual SLAM

Dissertation

Tobias Pietzsch

Technische Universität Dresden

Towards Dense Visual SLAM

Dissertation

zur Erlangung des akademischen Grades
Doktor rerum naturalium (Dr. rer. nat.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inf. Tobias Pietzsch
geboren am 25. Dezember 1975 in Rodewisch

Gutachter: Prof. Dr. rer. nat. habil Steffen Hölldobler
(Technische Universität Dresden)
Dr. Andrew Davison
(Imperial College London)

Verteidigung: 7. Juni 2011

Dresden, im November 2011

Acknowledgements

I want to express my gratitude to Prof. Steffen Hölldobler for supervising this work and giving me the opportunity and freedom to follow my own research interests. He has provided a very pleasant and stimulating work environment in these past years. I would also like to thank him for his recommendations on the manuscript.

I want to thank Dr. Axel Großmann, who has accompanied this work from the beginning. I am deeply grateful for his frequent encouragement and many inspiring conversations. During the preparation of the manuscript, his constructive comments have been invaluable.

I thank Jan Funke for many insightful discussions over coffee. He has been a great student, collaborator, and friend. I really enjoyed our work on the evaluation framework.

Finally, I would like to thank my parents for their constant support and encouragement. I could not have completed this work without it.

Contents

1. Introduction **1**
- 1.1. Simultaneous Localisation and Mapping . . . 1
- 1.2. Visual SLAM . . . 2
- 1.3. Towards Dense Visual SLAM . . . 3
- 1.4. Contributions . . . 4
- 1.5. Outline . . . 7

2. Related Work **9**
- 2.1. History of Visual SLAM . . . 9
- 2.2. Application of Stereo Cameras . . . 15
- 2.3. State of the Art . . . 16
- 2.4. Higher-Level Features and Dense Maps . . . 21

3. Preliminaries **27**
- 3.1. Vector and Matrix Notation . . . 27
- 3.2. 3D Coordinate Systems and Pose Transformations . . . 28
- 3.3. Rotation Parameterisations . . . 29
- 3.4. Homogeneous Representation of 2D points . . . 33
- 3.5. Perspective Projection and Camera Parameters . . . 33
- 3.6. Probability Theory . . . 37
- 3.7. State Estimation in Dynamical Systems . . . 42
- 3.8. The Bayes Filter . . . 43
- 3.9. The Kalman Filter . . . 45
- 3.10. The Extended Kalman Filter . . . 48
- 3.11. The Iterated Extended Kalman Filter . . . 51

4. The EKF Framework for Visual SLAM **55**
- 4.1. Introduction . . . 55
- 4.2. Assumptions . . . 56
- 4.3. Overview . . . 56
- 4.4. State Vector Parameterisation . . . 60
- 4.5. Process Model and Prediction Step . . . 62
- 4.6. Measurement Model and Update Step . . . 63
- 4.7. Feature Measurement Using Active Search . . . 66
- 4.8. Map Management . . . 74
- 4.9. Parameter Settings . . . 78

5. Evaluating Visual SLAM **81**
5.1. Introduction . . . 81
5.2. Related Work . . . 82
5.3. Framework Overview . . . 83
5.4. Repository . . . 85
5.5. Rendering System . . . 86
5.6. VSLAM Interface . . . 87
5.7. Evaluation System . . . 88
5.8. Usage Examples . . . 89
5.9. Discussion . . . 94

6. Efficient Point Feature Representation – Inverse Depth Bundles **95**
6.1. Introduction . . . 95
6.2. Related Work . . . 96
6.3. Point Feature Appearance and Measurement . . . 98
6.4. Inverse Depth Parameterisations . . . 102
6.5. View-Point Based Inverse Depth Parameterisation . . . 105
6.6. Inverse Depth Bundle Parameterisation . . . 111
6.7. Appearance Prediction and Measurement for Bundle Features . . . 114
6.8. Biased Measurements . . . 118
6.9. Experimental Evaluation . . . 122
6.10. Discussion . . . 147

7. Planar Features For Visual SLAM **151**
7.1. Introduction . . . 151
7.2. Related Work . . . 153
7.3. Direct Intensity Measurement Models . . . 158
7.4. Representing and Measuring Planar Features . . . 160
7.5. Fusing Measurement Information . . . 167
7.6. Iterative Measurement Process . . . 179
7.7. Initialisation of New Planar Features . . . 182
7.8. Remarks . . . 186
7.9. Experimental Evaluation . . . 191
7.10. Discussion . . . 218

8. Conclusion **221**
8.1. Summary . . . 221
8.2. Future Work . . . 222

A. Jacobians **225**
A.1. Introduction: Jacobians of Euclidean Measurement Model . . . 225
A.2. Jacobians of Basic Operations . . . 227
A.3. Jacobians of the Constant-velocity Process Model . . . 233
A.4. Jacobians of the View-Point Based Measurement Model . . . 235
A.5. Jacobians of the View-Point Based Feature Initialisation . . . 237

A.6. Jacobians of the Inverse Depth Bundle Measurement Model 238
A.7. Jacobians of the Inverse Depth Bundle Feature Initialisation 239
A.8. Jacobians of the Planar Measurement Model 240

1 Introduction

In this chapter, we will motivate our research goal and give an overview of the contributions made in this thesis. The background of this thesis is the field of visual Simultaneous Localisation and Mapping (SLAM), which concerns with simultaneously estimating the pose of a camera and a map of the environment from a sequence of images. Sections 1.1 and 1.2 provide an informal introduction; an in-depth review of relevant work will be given later. Traditionally, visual SLAM employs sparse maps comprising isolated point features, which facilitate robust localisation but are not well suited to advanced applications. Our goal in this thesis is to improve this map representation to allow a more dense description of the environment. This will be motivated in Section 1.3. An overview of our contributions towards this goal follows in Section 1.4. We conclude with an outline of the thesis in Section 1.5.

1.1. Simultaneous Localisation and Mapping

To form an internal model of the world from perceptions is often considered a key ingredient of true autonomy in intelligent agents. An internal representation of the world helps to understand the state of the world and how it will change in response to the agent's actions. Such understanding is the basis for informed decision-making and planning ahead. In particular, for autonomous navigation, the agent needs to maintain a map, i.e., an internal spatial representation of its environment. The agent also needs to know its own location with respect to the map. This is obviously necessary for the task of navigation but is also necessary for map-building itself. Keeping track of the location is a prerequisite for coherently updating and extending the map using sensory perceptions. In robotics, this problem is referred to as Simultaneous Localisation and Mapping (SLAM).

SLAM is a difficult problem, although to us humans it may seem simple at first. That is because we solve it routinely and often subconsciously in our daily lives. Using our senses we effortlessly navigate familiar environments. To appreciate the difficulty of SLAM, it is

helpful to consider larger scale problems where we actually use physical maps. Localising yourself with respect to a terrain map in unknown surroundings can be quite difficult. Creating a terrain map is an even more complex task that requires specialised tools and skills.

Endowing mobile robots with the ability to solve SLAM, even on a local scale, is a challenge that has occupied researchers for several decades now. The task for a robot performing SLAM is to build a map of an unknown environment and simultaneously localise itself with respect to this map. The information provided to solve this task are readings from unavoidably noisy sensors. Moreover, expected outcomes of actions can be employed, such as the expected location change after performing a motion command. Uncertainty is inherent both in the sensor readings and the predicted outcomes of actions. It has proven successful to make this uncertainty explicit through a probabilistic representation of the ambiguous knowledge about the state of world. Given this representation, probabilistic inference techniques can be used to integrate new information. Nevertheless, casting this approach into computationally feasible algorithms is difficult and requires carefully chosen approximations.

An important requirement for SLAM algorithms is to provide intermediate solutions. The map should be build incrementally in parallel to the operation of the robot. An estimate of the current location should be available at any time to aid decision making and navigation.

1.2. Visual SLAM

A variety of sensors have been used for SLAM. Odometry sensors such as wheel encoders provide estimates of 2D robot motion on even terrain. Time-of-flight sensors such as laser range finders and sonar sensors have been traditionally used to provide range measurements to objects in the environment. Early SLAM systems often considered robots moving on a plane in structured indoor environments, where wall segments and corners can be identified in planar laser scans. This effectively reduced the problem to a 2D setting.

SLAM has been maturing towards unconstrained 6 degree of freedom motion in general 3D environments. In this setting, cameras are currently the predominant type of sensor. Cameras provide rich visual information about the environment. Visual appearance of objects is often much more distinctive than, e.g., the distance field of a laser range scan. This eases the disambiguation of measurements and the association of measurement data to objects in the map. Cameras provide images at comparatively high frame-rates, which allows to put strong priors on the predicted frame-to-frame motion. They are inexpensive, light-weight, and consume little power.

All of this makes vision an attractive sensing modality for robotic SLAM applications. The ubiquity of cameras in many consumer devices also opens up new applications beyond robotics. For instance, the camera in a mobile phone might function as a passive localisation device. Another example is augmented reality, where a live image stream is augmented with artificial information. Virtual objects are overlaid on the video to give the illusion that they are present in the environment. To achieve this, the view-point relative to the scene must be known, i.e., localisation of the camera is required.

In this thesis, the term *Visual SLAM* refers to SLAM with a passive camera as the *only* sensor. The camera may be hand-held or otherwise attached to a human or moving system. A breakthrough achievement in this regard is the work by Davison (2003) who for the first time demonstrated real-time visual SLAM using only a hand-held monocular camera. This is achieved by continuously refining a map estimate using an Extended Kalman Filter (EKF).

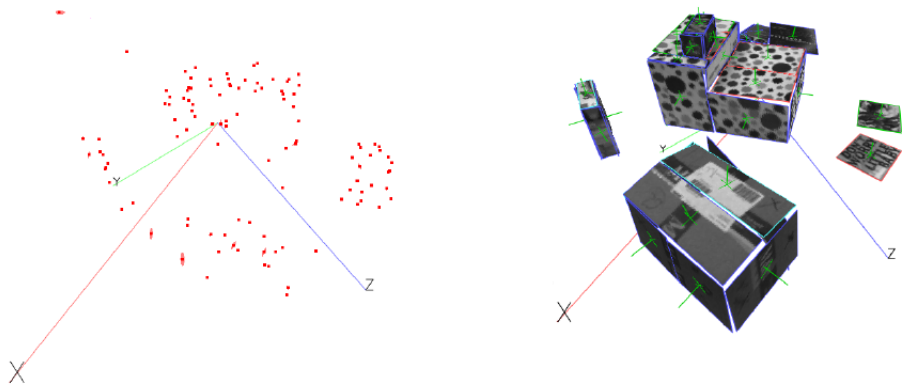
Despite the many advantages of camera sensors, visual SLAM is an extremely challenging problem. There is no active control over the movement of the camera, which makes it difficult to predict the camera motion between frames. Real-time processing of the images at high frame-rates is required for tracking the camera location under these conditions. The images are the only sensor information that is available. Because of the projective nature of a camera, depth is not directly observable in the images. Instead, 3D information must be recovered by establishing correspondences between image features. In monocular SLAM, these correspondences must be established over time. In this thesis we use a stereo camera, which additionally allows to establish correspondences between the left and right image of a stereo pair.

1.3. Towards Dense Visual SLAM

Visual SLAM has progressed considerably since the original work of Davison (2003). Many shortcomings and restrictions of early systems have been satisfactorily addressed. Appearance-based re-localisation methods allow to recover from tracking failure and to reliably detect the closure of large loops. Progress in estimation methods has brought a move towards large-scale maps beyond the limitations imposed by the single Extended Kalman Filter approach.

These advances have primarily focused on accurate and robust localisation performance. Comparatively little progress has been made on the map representation, though. The vast majority of today's systems still employ sparse point maps. Such maps comprise a small set of point features, which correspond to fixed 3D points in the environment and have a salient visual appearance that allows to identify and match them between camera images. From the localisation point of view sparse point maps are attractive. They allow to minimize the computational resources spent on image processing while providing sufficient information to keep track of the camera pose. However, as we will illustrate in the following, sparse point maps are of limited use as semantically meaningful representations of the environment.

Visual SLAM approaches a point of maturity where systems will be applicable in a wide range of real-life settings. Many application areas will benefit from, or even require, maps that comprise dense geometric information. Higher-level geometric information will improve the human-readability of the maps produced by a SLAM system which can be important for example in remotely controlled robotic scenarios. Consider the example shown in Figure 1.1. The figure shows two maps of the same environment: a sparse point map (a) and a partially dense map comprising textured surface segments (b). Without knowing the corresponding input images, it is difficult to give an interpretation of map (a). However, map (b) provides a fairly good impression of the structure of the scene.



(a) (b)

Figure 1.1.: A sparse point map (a) and a map of textured planar segments (b) of the same scene.

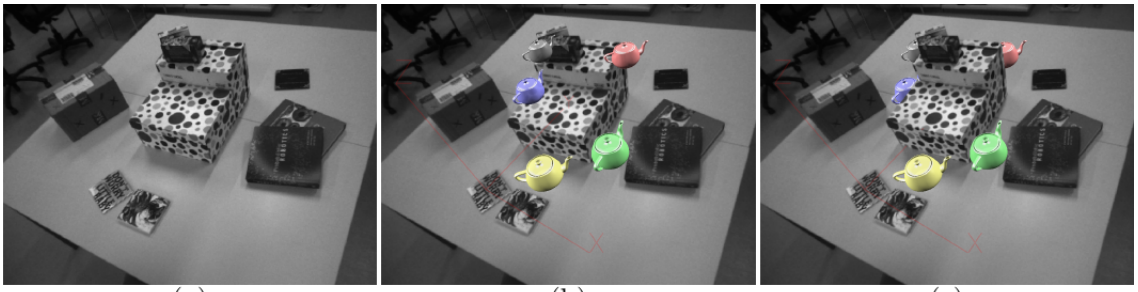
Another application that requires dense maps is real-virtual occlusion in augmented reality. Consider the example in Figure 1.2. Given accurate camera localisation, artificial objects can be inserted into a video stream such that they remain at a fixed position in the scene as the camera moves. If the camera pose is known, input images can be augmented with virtual objects that appear perspectively correct and attached to the real scene, cf. Figure 1.2(b). However, to achieve a more credible illusion, virtual objects should be occluded by real objects that are closer to the camera, cf. Figure 1.2(c). A point map is not sufficient to achieve this.

As a further example consider path planning in a mobile robot. This requires reasoning about free-space. The robot has to answer questions such as: "How can I travel from A to B without bumping into an obstacle?" A sparse point map lacks the required information for this type of task.

In general, sparse map representations are not adequate when we want to understand geometric relationships among objects in the scene, as well as predict or simulate interactions between them. If we want visual SLAM to move beyond tracking applications we need maps representing dense structure which allow geometric reasoning. Employing dense maps within the EKF framework for visual SLAM is the principal motivation of this thesis.

1.4. Contributions

In this thesis, we explore two broad directions towards a denser representation of the environment. First, there is the possibility to build maps that contain more points. We



(a) (b) (c)

Figure 1.2.: Real-virtual occlusion. Given the camera pose an input image (a) can be augmented with virtual objects that appear attached to fixed locations in the environment (b). For a more credible illusion, virtual objects should appear occluded by real objects in front of them (c).

propose a particularly efficient parameterisation of point features to reduce the size of the map representation. This allows fully correlated point maps that contain up to four times more features than previously possible. This technique can be employed both to construct spatially larger maps and to construct denser point maps. Ultimately, the usefulness of point maps is limited, though. Therefore, the second possibility that we explore is to go beyond point maps and use more descriptive features, such as line or surface segments. To this end, we propose a representation and measurement methodology for planar features. The measurement method relies on the ability to accurately predict the visual appearance of such features from varying camera view-points. The basis for this is already laid in the first part of the thesis, where particular consideration is given to appearance prediction of point features. For both the point and planar feature models we include a reference camera pose into the probabilistic map estimate. This reference camera pose describes the relative position of a keyframe image that gives a snapshot of a feature's appearance. A key ingredient to modeling the feature measurement processes in both the planar and point representations is the insight that observations must be understood as relative to this reference. We thoroughly evaluate our map representations on both simulated and real data. For the simulation we use computer-generated image sequences with exactly known ground truth. This method allows evaluation with respect to known ground truth under more realistic conditions than previously used in visual SLAM.

Specifically, with this thesis we make the following contributions:

An Evaluation Framework for Visual SLAM Algorithms. We propose a method for evaluating visual SLAM systems on synthetically generated image sequences. For this purpose, a complete and extensible framework is provided which handles image rendering, ground truth generation, and automated evaluation. This performance evaluation system was implemented in joint work with Jan Funke.[1] It is completely built on free software and is publicly available. The framework is used extensively for experiments in this thesis.

[1] At that time Jan Funke was a student whom I supervised. He contributed to the evaluation system partly as a student assistant and partly during his diploma thesis.

The motivation for this work was a lack of generally accepted performance measures, test frameworks, and ground truth benchmark problems for visual SLAM. Currently, systems are evaluated by visual inspection of the results on recorded image sequences, and/or measuring accuracy on simulations of simplified point-cloud-like environments. However, both approaches have drawbacks. Recorded sequences lack ground truth while simulations tend to oversimplify low-level aspects of the problem. Evaluation on rendered image sequences, on the other hand, combines advantages of both approaches.

Reliable and accurate ground truth is available in rendered image sequences. However, rendered images allow to examine details below the level of abstraction of point clouds. Many unmodeled effects occurring in real sequences can be convincingly emulated in rendered images. In this way, simulation conditions can be moved much closer to the realism of recorded real imagery while still having ground truth available.

A novel aspect of the proposed framework is adaptive generation of ground truth. The evaluation framework does not interfere in any way with the system under evaluation. In particular, the visual SLAM system automatically selects visual features in the images. The ground truth map is calculated by tracing the 3D locations for these selected features. The SLAM system is not restricted in its choice of features which means that the effects of various map management heuristics are not ignored in the evaluation.

This work has been peer-reviewed and published in (Funke & Pietzsch, 2009a).

The Inverse Depth Bundle Parameterisation. We propose a novel point feature representation that is more efficient than previous parameterisations, in the sense that the state size per feature is smaller. When combined with an appropriate heuristic for new feature initialisation the inverse depth bundle parameterisation can reduce the state dimensions per feature to less than half the size of the straightforward Euclidean and less than a quarter of the size of the popular unified inverse depth parameterisation by Montiel et al. (2006). This means that fully correlated maps with more features than before can be updated in real-time. It allows to build maps that are more densely populated or span larger environments.

An additional goal in developing this feature model was to allow accurate feature appearance prediction using homography warping. We analyse in detail the generative process that leads to image measurements. This analysis indicates that the initialisation camera pose for every feature should be part of the state. Moreover, it suggests the use of a one-parameter representation for features (with respect to the initialisation pose). Such parameterisations have been criticised by some authors because they neglect bias introduced by the initial measurement. We present an empirical analysis of these bias effects. Experiments are carried out on artificial image sequences. This allows for a fair evaluation that is impossible with point cloud simulations.

Parts of this work have been peer-reviewed and published in (Pietzsch, 2008a).

A Representation and Measurement Methodology for Planar Features. By using landmarks that are more descriptive than point features, such as surface segments, larger parts of the scene can be represented in a compact form. This minimises redundancy and might allow applications such as object detection and path planning. We propose a probabilis-

tic map representation for planar surface segments. These planar features are measured directly using the image intensities of individual pixels in the camera images. In this way, the information provided by changes in feature appearance due to changing view-point is directly used to improve the state estimate. Linearising pixel intensity measurements (as required by the EKF) is problematic because image functions are often highly nonlinear. This would severely restrict the amount of image motion that can be handled robustly. We propose an iterative measurement update step to obtain the same robustness in this respect as for point feature matching. The update is initialised using standard correlation search known from point features. Then the result is iteratively refined using intensity measurements. Experimental results show robust camera tracking using planar features and increased accuracy in comparison to traditional point features, even when only a single planar feature is used.

Parts of this work have been peer-reviewed and published in (Pietzsch, 2008b).

A Method for Integrating High-Dimensional Measurements in the EKF Update. A distinguishing property of the planar features described above is that the measurement vector is potentially very large. The dimension of the measurement vector depends on the size of the area where a planar feature is visible. Measurements comprising thousands of pixels are easily possible. This leads to serious performance problems if these measurements are used straightforwardly because the EKF update has cubic complexity in the size of the measurement vector. We present a principled and efficient way of reducing high-dimensional measurements to fused measurements of bounded size. The cost of this reduction is linear in the measurement. Thus the complexity of the update step is reduced from cubic to linear in the size of the measurement vector. This reduction operation is derived using the duality of the Kalman Filter and the Information Filter. The exploitation of this connection is a novel theoretical result.

1.5. Outline

In Chapter 2 we review important previous work to give an overview of the history and state of the art in visual SLAM. In particular, relevant related work towards dense reconstruction is reviewed and classified, providing the background for this thesis. More detailed discussions of related work are presented in the chapters comprising the body of this thesis with respect to the specific contributions.

In Chapter 3 we establish the notation and mathematical background for the rest of the thesis. This includes transformations of 3D points as well as their projection in camera images. Basic concepts from probability theory are reviewed leading to the Extended Kalman Filter (EKF) as the underlying inference mechanism of many visual SLAM systems.

In Chapter 4 we present a generic but complete implementation for visual SLAM based on the EKF. The system is in many respects similar to the original system of Davison (2003). Some extensions are made, concerning the use of a stereo camera and the identification of erroneous measurements using the data association method by Neira & Tardos (2001). Despite its lack of the bells and whistles required for long-term and large-scale localisation, the system serves well for testing the map representations developed in this

thesis. At the same time, the description of the implementation provides an introduction to the technical concepts and challenges of visual SLAM.

Chapters 5 to 7 comprise the main body of this thesis. They present the contributions outlined in Section 1.4 above. In Chapter 5 we introduce an evaluation framework for visual SLAM systems. In Chapter 6 we develop the Inverse Depth Bundle map parameterisation that allows an efficient representation of point features. In Chapter 7 we discuss a parameterisation for planar features and a direct measurement method that is tightly integrated into the EKF estimation framework. Moreover, we propose a general approach to handling high-dimensional measurements.

The thesis concludes in Chapter 8 with a summary of the contributions and an outlook to future work.

2
Related Work

Over the last decades SLAM in general has been a very active field of research. In recent years, the use of visual sensors has become increasingly attractive for this task, especially with recent advances in hardware and computational power that allow real-time processing of the enormous amount of data available from camera image streams. Remarkable progress has been made towards visual SLAM systems that function accurately and robustly in extensive environments over long periods of time.

The purpose of this chapter is to put the work presented in the thesis into the context of the current state of the art in visual SLAM. We give a broad overview of historic and recent work in the field. This overview will be supplemented by focused in-depth discussions of related work in Chapters 5 to 7, pertaining to the specific topics of those chapters.

The remainder of this chapter is structured as follows. A brief history of visual SLAM is given in Section 2.1. We will review the major contributions from the fields of both robotics and computer vision. Because the system developed in this thesis employs a stereo camera, we focus on approaches using stereo vision in Section 2.2 and discuss the differences to monocular vision. The current state of visual SLAM is reviewed in Section 2.3. We look at the issues that must be addressed to achieve large-scale and robust localisation. We discuss examples of recent approaches. Finally, in Section 2.4 we turn to the specific problem tackled in this thesis, namely the construction of maps with dense or higher-level information. We review in depth the recent advances in this area, providing the context for our own contributions.

2.1. History of Visual SLAM

The *Simultaneous Localisation and Mapping* (SLAM) problem originated in robotics. Here, the task for a mobile robot is to build a map of an unknown environment and simultaneously localise itself with respect to that map. Visual SLAM tackles this task

with cameras as the primary sensor. In this thesis we subscribe to an even stricter definition of visual SLAM, namely that the camera is the *only* sensor, and that the camera is freely moving in space. There is no active control over the camera motion. For instance, the camera might be held in the hand of a human operator.

There is a natural overlap with the field of computer vision, where the same problem is referred to as *Structure from Motion* (SFM). The history of visual SLAM is firmly rooted in both fields. Key ideas have been developed independently in both fields in parallel. Today we see a convergence and unification of terminology and methods. The full generality of the problem is becoming clear only recently.

Traditionally, robotics and computer vision have had a slightly different focus in the approaches to the problem. In robotics, the attention is strongly on sequential techniques that provide estimates online using the data so far acquired. A mobile robot needs an up-to-date estimate of its position and the environment map to make informed decisions. The map is incrementally built during exploration of the environment instead of constructed in retrospect from the data collected during a run. In contrast, computer vision has often approached the problem as one of batch optimisation.

In the following, we review seminal work from robotics, in Section 2.1.1, and computer vision, in Section 2.1.2, leading up to the first real-time vision-only systems.

2.1.1. Simultaneous Localisation and Mapping

To rationally interact with a complex environment, a robot has to maintain an internal model of the outside world. Due to the noisy nature of real world sensors and actuators there is inherent uncertainty. A breakthrough insight has been that this unavoidable uncertainty necessitates its explicit representation and handling in the estimation machinery.

The seminal work of Smith et al. (1986, 1988) introduces the *stochastic map* to represent relationships among spatial entities, making explicit the inherent uncertainty. The proposed representation is a probability distribution over the uncertain variables, where the distribution is estimated by its mean vector and covariance matrix. They show how to carry out various operations on the map, such as moving objects, adding constraints obtained from measurements, and predicting measurements and their utility. The operations of moving the robot and integrating measurement constraints amount to the prediction and update equations of the Extended Kalman Filter (EKF). In the prediction step, a motion model is used to estimate the evolution of the robot position over time. In the update step, a measurement model is used to refine the stochastic map (including the robot position).

A similar formulation and one of the first practical implementations is presented by Moutarlier & Chatila (1989). They apply the stochastic map and the EKF to mobile robot mapping. A sparse line map is built using measurements by a 2D laser range finder.

Following the above and similar approaches, the EKF became *the* most popular technique to address SLAM. A fundamentally important characteristic of the approach is the maintenance of a full covariance matrix. This naturally encodes correlations between map entities. It turns out that this property is crucial to performing SLAM with noisy information, as it facilitates the building of consistent maps. The importance of maintaining correlations has been repeatedly stressed. For example, Castellanos et al. (1997) exper-

imentally compare the results of an approach that assumes independence between map entities with one that models correlations. They apply an EKF to perform SLAM using odometry and laser range finder sensors. They build a 2D map of an indoor office environment using straight line segments that are extracted from the laser measurements. Their experiment shows that assuming independence between map features causes estimates to be overly optimistic. In turn, this leads to low compatibility between observations and the map.

Vision has been applied among other sensors in robotics. It has become an increasingly popular sensing modality for a variety of reasons. Cameras are inexpensive, lightweight, low-power, and passive sensors. Image streams from a camera provide massive amounts of information when compared to other sensors such as sonar or laser range finders. Growing computing power allows to utilise more and more of this information. The following are some examples of robotic SLAM systems employing vision sensors.

Neira et al. (1997) present a SLAM system that builds indoor maps using monocular vision. The motion of the robot is restricted to a 2D plane, and odometry sensors provide an estimate of the robot trajectory. The system is built on a stochastic map and the EKF. Vertical edges are used as features in the 2D map. In man-made indoor environments, vertical edges are often prominent features. Under the constraint of planar motion, vertical edges correspond to vertical lines in the camera image. The resulting technique may be characterised as "2D monocular vision".

Se et al. (2002) make use of a trinocular stereo camera. Similar to the work above, the robot is restricted to 2D planar motion and uses odometry to provide trajectory estimates. However, the system builds a true 3D map using SIFT key-points (Lowe, 2004) as features. Map features are localised using trinocular stereo vision as follows. First, the images are exhaustively searched for SIFT key-points. Correspondences across the three views are established using restrictive matching criteria on key-point scale and orientation as well as spatial configuration. Triangulation of successful matches yields a 3D position. The map is constructed as a database of 3D point locations with associated SIFT descriptors. By matching current features to features in the database, a robot pose estimate is obtained using least squares fitting. The system does not employ a full stochastic map. Instead, the positions of database features are estimated using one Kalman Filter per landmark. Thus, no correlations between features are maintained. However, the authors still achieve reasonable results for a room-sized environment.

Davison & Murray (1998) present a SLAM system for a robot equipped with an active stereo head. The stereo head has four axes of freedom allowing the cameras to fixate on specific points in the environment. The robot is assumed to be moving on a plane. The system builds a sparse stochastic map of persistent visual features that is maintained using an EKF. Candidate features are found by the Shi & Tomasi (1994) corner detector and epipolar matching. Small pixel patches around these corner points serve as map features. Measurements of these are made by correlation search in the images. The authors present results where the robot automatically navigates an unknown environment. In a further experiment, they eliminate the cross-covariance between map features. This leads to an underestimation of uncertainty and failure to successfully re-acquire features after a period of neglect. This corroborates the results of Castellanos et al. (1997), again emphasising the importance of maintaining full correlation information.

One notable aspect of the system of Davison & Murray (1998) is the active search approach to measurement. The authors note that the EKF approach allows to predict image measurements and their expected uncertainty. The predicted measurement is used to actively fixate the stereo head on the expected feature location. The predicted uncertainty allows both to restrict the image search as well as to decide which feature to measure next. The projection of the measurement uncertainty into the images leads to elliptical search regions in the image where the feature will be found with high probability. Limiting the correlation search to these areas reduces both, the computational cost and the chance of mismatches. The selection of the best feature to measure next is also based on the predicted measurement uncertainty. In terms of information gain it is most advantageous to measure the least certain feature.

Davison (2003) extends the approach above to a single hand-held camera. This scenario is much more challenging in several ways. First, there is the passive nature of the system and the absence of odometry information. This means that only a basic motion model is available and localisation uncertainty grows rapidly. Second, the use of a monocular camera instead of the stereo head means that 3D measurements are not directly available from the sensor. Instead, this is an instance of *bearing-only* SLAM, where 3D information arises only through the motion of the sensor and has to be aggregated over time. Nevertheless, much of the underlying formulation is inherited from the earlier system of Davison & Murray (1998). A sparse map of point features is maintained using an EKF. Image measurements of features are obtained using correlation search. The motion model is adapted to reflect the basic assumption that the camera continues to move with constant velocity under the absence of external forces. The unknown external forces are modeled as noise which leads to quickly increasing camera uncertainty. This absolutely necessitates real-time operation where every camera image must be used to sufficiently constrain the camera location.

An important insight is that the active search approach carries over to the passively moving camera. No longer can features be actively fixated. However, the predicted image measurements and uncertainties can still be employed to guide the measurement process. Rather than controlling the motion of the active stereo head, this information is now used to decide on which image areas the limited processing power should be focused. As before, the uncertainty is also used to decide which features to measure next. Moreover, it is employed to restrict the search area for a feature to an elliptic image region.

Because of the projective nature of the sensor, full 3D information is not available from a single feature observation. Newly selected features lack depth information and thus can not be directly inserted into the stochastic map. This is handled by initializing new features in a separate particle filter until their depth is sufficiently well-constrained to insert them into the map.

Davison (2003) is undoubtedly a seminal paper, presenting the first real-time system to achieve SLAM with a hand-held camera. Most importantly perhaps, it attracted attention from both the robotics and computer vision community, marking a point of convergence of ideas from robotic SLAM on the one hand and Structure from Motion in computer vision on the other hand. The work also serves as a starting point for this thesis. The SLAM framework developed in Chapter 4 of this thesis is heavily based on Davison's system.

2.1.2. Structure from Motion

In computer vision, the related problem of Structure from Motion (SFM) is considered. Here, the task is to reconstruct the camera parameters and scene structure from a set of images. In Structure from Motion, batch solutions have been popular because they achieve the most accurate results by considering all images simultaneously.

An excellent introduction and overview of classical Structure from Motion methods is given by Hartley & Zisserman (2000). A well-established procedure taken by most approaches is the following. First, 2D correspondences among the input images are established. The images are exhaustively searched for interest points, e.g., using detectors by Harris & Stephens (1988); Shi & Tomasi (1994); Rosten & Drummond (2006). Putative feature correspondences between images are established using local correlation measures or feature descriptors (Matas et al., 2002; Lowe, 2004). Robust methods such as RANSAC (Fischler & Bolles, 1981) are then used to obtain consistent sets of correspondences and build an initial reconstruction. Finally, this reconstruction is refined using constrained nonlinear optimisation. This last step is referred to as *bundle adjustment*, alluding to the bundles of rays that connect 3D feature and camera positions.

The work of Fitzgibbon & Zisserman (1998) serves as an example of a typical batch approach. The authors tackle the problem of reconstruction from uncalibrated image sequences. They propose a hierarchical approach. Interest points are matched between pairs of views using the epipolar constraint to obtain robust matches. Then a projective reconstruction is obtained for every triplet of consecutive images. The triplets are then aligned into subsequences. Subsequences in turn are aligned into the full sequence, possibly taking into account additional overlap constraints in image sequences that are known to be closed. The final solution is refined using bundle adjustment, which is also employed at various intermediate stages of the algorithm.

The solutions obtained using batch methods are superior in accuracy to those obtained by sequential filtering methods, because of the bundle adjustment that is invariably the last step in almost all batch methods. Bundle adjustment treats SFM as a large parameter estimation problem, the parameters being the camera parameters of all the images and the structure of the scene, i.e., the 3D coordinates of interest points. Nonlinear optimisation is used to jointly optimise all parameters with respect to some cost function. The cost function is often a robust function based on the re-projection error, i.e., the difference between the image observations of interest points and the expected projection using the parameters. Bundle adjustment has a long history itself, originating from photogrammetry. A comprehensive survey of bundle adjustment is given by Triggs et al. (1999).

Traditionally, in real-time SLAM, filtering methods have been favoured over bundle adjustment, despite its superior accuracy. Although bundle adjustment can be much more efficiently implemented than a general nonlinear optimisation method by exploiting the sparse structure of the problem, the fact remains that computational cost grows without bounds with increasing length of the image sequence. However, as we will discuss later, modern visual SLAM systems increasingly rely on bundle adjustment techniques.

Batch methods, as described above, put a different focus on the problem than sequential SLAM approaches. In SLAM, we explicitly operate on *sequences* of images with the implied temporal and spatial correlation between images. Such information is often neglected

in batch approaches, i.e., they do not employ an explicit motion model. There is often no assumed ordering or spatial adjacency relation among the images. However, sequential approaches to SFM have been explored in the computer vision literature as well. These often use filtering techniques similar to the stochastic map techniques from SLAM.

Broida et al. (1990) present one of the earliest sequential SFM approaches. A number of feature points on an unknown but rigid object is tracked to estimate its motion in a sequence of images.[1] They construct a state-space model which parameterises both, the structure and the motion. Motion is described using a constant velocity motion model, similar to Davison (2003). An Iterated Extended Kalman Filter (IEKF) is used to update the model with image observations of the feature points. The model is initialised with a batch estimate over a few frames. The authors argue that the recursive, sequential approach is advantageous because an arbitrarily large number of images can be used. The issue of extracting and matching features is not addressed in the work. In the experimental results image measurements are made manually. However, the authors note that the predictive capabilities of the filter can be used to aid in the measurement process, anticipating a core idea of active search. They propose to use the IEKF predictions with uncertainty to restrict search regions for future feature matchings. They also expand on the use of the filter estimate for Maximum Likelihood data association.

Azarbayejani & Pentland (1995) present a similar recursive approach to SFM from image sequences. Their method is based on the EKF. The system dynamics are described by a constant position model. A main contribution is partial self-calibration, i.e., the camera's focal length is estimated along with the structure and motion. They introduce a novel parameterisation that is optimised for focal length estimation. The motion in the direction of the camera axis is represented as the ratio of depth and focal length because this quantity remains estimable for long focal lengths, whereas the depth does not. The coordinate origin is fixed at the image plane instead of the camera's projection center. This is supposed to decouple the representation of structure and focal length. Each feature is represented in the state-space by its unknown depth along a perspective ray that is assumed known. This one parameter feature representation is closely related to the bundle representation that we propose in Chapter 6 of this thesis.

More recently, Jin et al. (2003) present a real-time system for SFM reconstruction from a monocular camera. Their system is also based on an EKF. In contrast to the approaches mentioned above they explicitly address the handling of occlusions, i.e., they remove features that become occluded and initialise new features during the course of the sequence. Every new feature goes through an initialisation phase in its own separate filter. After a probation period the new feature is transformed to the inertial coordinate frame and inserted into the main filter.

The estimation framework and parameterisations of the sequential approaches described above are very similar to those of sequential SLAM systems such as (Davison, 2003). The crucial difference is that SLAM systems use a map of *persistent* features that remain in the state-space representation while they are temporarily occluded. Those features can be re-acquired as they become visible again, which allows for long-term operation without unbounded accumulation of error drift.

[1] Of course, this corresponds to estimating the motion of a moving camera in a static scene.

2.2. Application of Stereo Cameras

In this thesis, we employ a stereo camera as the sensor in our visual SLAM system. Both, stereo and monocular systems have been considered in the literature. The availability of monocular cameras in mobile phones and other hand-held devices makes monocular SLAM a relevant and attractive research goal. However, monocular SLAM is certainly the more difficult instance due to the non-observability of scale in the environment. The use of a stereo camera provides an instant measure of scale through the baseline between the two eyes of the camera. Especially in large-scale robotic systems, stereo cameras are popular because of the increased robustness they provide. Visual odometry approaches (e.g., Nistér et al., 2006) provide low-drift trajectories over many kilometers using stereo vision.

One way to employ a stereo camera is to treat it as a 3D measurement device such as a laser scanner. 3D measurements are provided by the extraction of matches between the images, followed by triangulation of the measurement rays. The approach of Se et al. (2002), which has been described above, falls into this category. Here, SIFT features are matched in a trinocular camera and triangulated to provide a 3D measurement.

Lemaire et al. (2007) present both, a monocular and a stereo approach, to visual SLAM. They keep only few long-term landmarks, in order to be able to perform loop-closures. Instead many volatile local features are added to the map that are removed as soon as they disappear from the image. In a direct comparison, the monocular approach shows slightly improved accuracy over the stereo approach. This is due to the fact that fewer measurements are available for the stereo case: Besides matching features temporally across frames, they also have to be matched between left and right images. This means that fewer matches can be established. On the other hand, the stereo system shows better consistency, i.e., under-estimation of errors seems to be a more serious problem in the monocular case. The stereo camera is treated as a 3D measurement devise, similar to the approach above.

Approaching stereo in this way has the disadvantage of the *dense fog* effect: The need for explicit triangulation of 3D measurement coordinates leads to a limited range of 3D observability. This means that remote objects can not be considered. The bearing information that very distant features could provide is ignored.

Solà et al. (2007) note that instead it is beneficial to treat a stereo camera as simply two monocular sensors. Consequently, bearing-only techniques are employed, albeit in conjunction the observations from both cameras provide instant depth information for some of the features. In particular, bearing-only techniques can consider features at infinity. These features serve an important function to provide compass-like global orientation measurements.

Paz (2008) presents a stereo visual SLAM system that employs a two-way strategy. Nearby features, for which depth is observable, are initialised as Euclidean 3D points. Distant features use the inverse depth parameterisation (Montiel et al., 2006), which is a well-known monocular approach to deal with the unobservable depth when initialising new features.

In this thesis we subscribe to the same philosophy as Solà et al. (2007). We treat the stereo camera as a trivial extension of the monocular case. A feature observation consists

of making monocular feature measurements in both the left and right image of the stereo camera independently. The disparity between both measurements corresponds to the inverse depth of the feature, which is automatically exploited in the EKF update. For instance, this helps in the initialisation of new features which benefits from the disparity measurement provided by the stereo camera. In conclusion, let us emphasize that the techniques developed in this thesis are equally applicable to both monocular and stereo scenarios. The use of a stereo camera here should be mainly seen as a means to provide additional robustness and simplify feature initialisation. The depth of nearby features can be directly inferred using the known baseline distance between the eyes of the stereo camera. This makes the overall scale of the reconstruction observable, which also eases experimental evaluation.

2.3. State of the Art

Visual SLAM has progressed considerably in recent years. Many shortcomings of the early systems have been addressed and satisfactorily solved. The field is rapidly maturing to the point where we will see robust, large-scale systems employed in real-world applications outside of the lab. Advances have been primarily made with respect to localisation. Robust detection of measurement failures and the ability to recover from total tracking failure provide stable localisation performance. Advances in estimation techniques allow to build consistent large-scale maps that facilitate operation in extensive environments. In this section we give an overview of recent work, organized by the issues that are addressed.

Comparatively little effort has gone into improving the map representations towards dense structure and higher-level semantic information. Most current systems still employ sparse point maps. We will review work on improved map representations in Section 2.4. We believe that this presents the next major challenge for visual SLAM.

2.3.1. Robust Data Association

Before sensor readings can be used to localise or improve the map, data association has to be solved. This is the task of establishing correspondence between raw sensor readings and map features. In visual SLAM, data association is a relatively good-natured problem. This is because features from a camera are endowed with rich visual appearance information which can help to disambiguate camera-to-map feature pairings.

Davison (2003) and similar systems solved data association on a per feature basis. Within a restricted search region, the image point with the most similar appearance to a given map feature is considered as the match. Because features are visually distinctive and the search regions are rather small, this yields excellent results most of the time. However, even very few mismatched measurements can lead to corrupted maps and subsequently to complete tracking failure, as recently illustrated by Clemente et al. (2007).

Neira & Tardos (2001) develop a data association technique that considers the *joint compatibility* of measurement–map feature pairings. Instead of deciding data association per feature independently, the joint compatibility refers to the set of pairings as a whole. This puts to use the correlations between predicted feature measurements. The authors propose an exhaustive search algorithm for the jointly most likely data association, the

Joint Compatibility Branch and Bound (JCBB) algorithm. JCBB has been used with great success in several visual SLAM systems (e.g., Clemente et al., 2007). It is also employed in the system presented in Chapter 4.

In Structure from Motion, variants of the RANSAC procedure (Fischler & Bolles, 1981) are usually employed. Rather than map-to-image association, the problem addressed is often image-to-image association. RANSAC stands for RANdom SAmple Concensus. In RANSAC, a minimal subset is sampled randomly from the set of putative feature correspondences. Minimal means that the set contains just enough correspondences to estimate the parameters we are interested in. For instance, 5 point correspondences are required to estimate the relative pose between two cameras. The estimated parameters provide a hypothesis, whose support is found by the total number of correspondences that agree with it. This sampling procedure is repeated for a number of times, and the hypothesis with the highest support is used to determine outliers, i.e., erroneous correspondences which do not agree with the hypothesis.

More recently, Chli & Davison (2008, 2009) presented the *active matching* approach, which is a logical continuation of the active search paradigm. Active matching is a departure from the two stage process used in JCBB or RANSAC, i.e., first obtaining candidate feature measurements and afterwards deciding data association. Instead, image processing, i.e., feature matching, is put into the loop of the search for a jointly compatible set of measurements. Feature searches occur one by one, where the next measurement attempt is decided based on expected information gain. Each feature search yields zero or more matches which are factored into a Sum-of-Gaussians representation of the multiple hypotheses for global data association. Each feature match carries information about the location of other features which further restricts the image search regions. The method was shown to search on average about eight times less pixels than the JCBB approach.

2.3.2. Loop Closing and Re-Localisation

Loop closing refers to the situation of re-entering a part of the map that has not been visited for some time. The stochastic map employed in a visual SLAM system handles closing of small loops automatically. Using the approximate knowledge of map and camera pose, it is easy to predict the image locations of features that re-enter the camera image after a short period of occlusion. For large loops, however, this is not sufficient. If a map area is re-entered after longer periods of exploration accumulated errors are too large to reliably restrict feature locations. Specialised loop detection algorithms are needed that recognise already visited parts of the map.

As noted by Eade & Drummond (2008) a closely related problem is re-localisation. Even the most robust system will sometimes loose track of the camera location because the underlying assumptions are violated. This may occur for example during phases of complete occlusion of the camera or because of unexpected rapid accelerations. Then the camera needs to be re-localised in the existing map. Eade & Drummond (2008) address both issues in a unified framework. Their system maintains a graph of small submaps. Nodes in the graph, i.e., submaps, are connected if they share some features. The shared features induce spatial constraints between connected nodes. Every node in the graph is endowed with a histogram of visual words observed while the camera was in

that node. SIFT-like descriptors are collected and quantised into a vocabulary of visual words that is learned online. Words occurring in the current image are then matched to the words expressed in any of the nodes producing a ranked list of visually similar nodes. The highest-ranking nodes are candidates for loop closure. A consistent set of feature correspondences between each candidate and the current node is sought using RANSAC. If enough correspondences are found, the candidate becomes a potential loop closure. After a trial period, an edge between nodes is added to the graph and the loop is closed. Re-localisation is treated as a special case. If the camera is lost, SLAM commences in a new disconnected component of the graph. A detected "loop closure" may connect the new component to the old map, thus re-localising the camera.

Williams et al. (2007) extend EKF based visual SLAM by a re-localisation module. The method is based on fast feature classification. Their key-point recognition algorithm adapts the randomised trees classifier of Lepetit & Fua (2006) for the requirements of real-time SLAM. Their classifier is trained online during the SLAM run. The classifier is modified to return multiple hypotheses and tuned towards high recall rate in order to perform well with the sparse set of features in SLAM. If the camera is lost, a key-point detector (Rosten & Drummond, 2006) is run on the full camera image. Potential matches between key-points and map features are established by the trained classifier. The classifier is fast but at the cost of a high false positive rate. Therefore, RANSAC is used to robustly find the pose of the camera from the putative correspondences. The approach is shown to allow real-time re-localisation in a map of 70 features.

Williams et al. (2008) extend this method to loop detection. The system is based on the hierarchical SLAM framework of Estrada et al. (2005). A sequence of independent submaps are built using the EKF. The re-localisation method described above is used to detect re-observation of previous submaps.

2.3.3. Estimation Methods and Large Scale Systems

Currently, the EKF is widely used in visual SLAM. It has been shown to work well for the typical office-sized laboratory domain. As the size of the environment and hence the map grows, two limiting factors of the EKF approach become apparent. First, it suffers from linearisation errors. Alternative filters, such as the Iterated EKF, the Unscented Kalman Filter, or Rao-Blackwellized Particle Filters (FastSLAM) have been considered in the literature as alternatives for improving local linearisation properties. However, ultimately the problem is inherent in all filtering approaches. They summarise past measurements into a state estimate, and linearisation decisions made in the past cannot be revised.

The second problem is the computational cost of EKF SLAM, which scales quadratically with the map size. This limits real-time performance to rather small maps.

A solution to this latter problem has been proposed independently by Guivant & Nebot (2001) and Knight et al. (2001), who refer to it as the Compressed EKF and Postponement, respectively. In both methods, a constant-sized subset of the map is updated based on current measurements. Updates to the remainder of the state vector are accumulated into auxiliary matrices. These can be used to perform a full update later. The full update is still as expensive as in the EKF but it is carried out only occasionally.

Submapping approaches address computational cost as well. Moreover, linearisation problems can be alleviated. The general approach is to build multiple local submaps of a bounded size and arrange these into a full map at the global level. Because the number of features in a local map is bounded, local mapping can be achieved in constant time. Because local uncertainty is relatively small, the linearisation errors remain small as well. Examples of submapping techniques in visual SLAM are given in the following.

Estrada et al. (2005) propose the Hierarchical SLAM framework. Here, local maps are built using the EKF. When a new local map is started, the reference coordinate frame for the new map is the current robot pose. Thus the final robot pose in the previous map provides a constraint which links the maps on the global level. New maps are started from scratch, thus providing conditional independence between submaps. Upon loop detection, the two involved submaps are fused using local map joining. On the global level, nonlinear optimisation is used to resolve the inter-map constraints. This method has been applied to visual SLAM by Clemente et al. (2007) and Williams et al. (2008).

Piniés & Tardós (2008) present a submapping approach using conditionally independent submaps. In contrast to previous approaches – such as the one discussed above – submaps are allowed to share information. Thus, instead of starting each new submap from scratch the new submap may be initialised using the robot pose and some features from the previous submap. Submaps built in this way are conditionally independent given the shared variables. Information can be propagated between maps in constant time, assuming that the shared state between submaps is bounded by a constant. Then the system achieves constant time exploration and linear time loop closure without introducing any approximations with respect to the full EKF solution. The assumption limits the system to relatively simple trajectories, though. Upon loop closure, the revisited features have to be copied into all submaps along the loop. Whenever a previous submap is revisited, a shared robot state is introduced between the submaps. More recently, the approach is extended to more complex trajectories by Piniés et al. (2009). They build a graph of conditionally independent submaps and information is propagated along a spanning tree over the graph. In experiments with a complex scenario comprising many loops they achieve approximately linear performance.

Another approach is provided by bundle adjustment, i.e., joint nonlinear optimisation of all camera and feature parameters. The complexity of bundle adjustment is linear in the number of features. However, it is cubic in the number of poses. More importantly, a straight-forward implementation would imply keeping all past camera images and optimising over all past poses in every step. Computation time thus grows without bounds as the length of the image sequence increases. Until recently, bundle adjustment was shunned in real-time SLAM because of these computational demands. This trend is reversing, though, as discussed in the following paragraphs. Current developments in visual SLAM are moving towards these optimisation methods.

Local bundle adjustment over the previous few frames provides accurate local estimates of structure and motion. Such approaches are referred to as *visual odometry*. Nistér et al. (2006) illustrate that such methods can provide low-drift estimates of trajectories of several kilometers. However, such approaches miss a key ingredient of SLAM: a long-term persistent map. Thus, indefinite drift-free localisation is impossible.

Another solution is to optimise the map over a sparse set of past keyframes and use

intermediate frames for localisation only. The Parallel Tracking and Mapping (PTAM) system of Klein & Murray (2007) is an impressive system using this technique. They split the tasks of mapping and localisation into two separate processing threads. The localisation thread runs at camera frame-rate using a fixed map that is periodically updated by the mapping thread. The mapping thread estimates the map using bundle adjustment of a sparse set of keyframes. This mapping procedure is run asynchronously in the background. Global bundle adjustment is restarted as new keyframes are added and may take tens of seconds to complete for larger maps. Typical maps comprise thousands of point features.

The recent analysis of Strasdat et al. (2010) compares filtering and sparse keyframe bundle adjustment. The authors argue that these approaches can be considered as two alternative ways of simplifying the full SLAM problem. Filtering approaches aggregate information from all frames of the sequence. However, computational cost restricts the map size and consequently, potential image features must be disregarded. Sparse keyframe approaches can handle large numbers of features. However, computational cost restricts the number of keyframes, and information from intermediate frames must be disregarded. Strasdat et al. (2010) empirically compare both approaches and conclude that keyframe bundle adjustment has a better accuracy per processing time ratio for most applications.

Konolige & Agrawal (2008) present the *frameSLAM* system which enables large-scale robotic mapping using a stereo camera and optionally an inertial measurement unit as sensors. The name refers to the sparse set of visual frames that form the map representation. Here, a frame corresponds to a pose of the camera and the associated stereo image. That is, the map consists of a set of past camera poses. The 3D environment structure, i.e., landmark positions, is not represented in the map.

Point feature correspondences between images induce constraints between the corresponding frames. Representing and bundle adjusting the full set of frames is infeasible. Therefore, Konolige & Agrawal (2008) propose to marginalise feature correspondence constraints, as well as intermediate frames from the full system. This marginalisation requires linearisation of the inter-frame relations. A novel contribution of frameSLAM is to lift the resulting local linear constraints to global non-linear constraints. After the marginalisation, a sparse skeleton graph of constraints between frames remains which is solved using standard nonlinear least squares optimisation. The system is demonstrated on sequences of tens of thousands of images over trajectories of 10 km length.

A different approach to large scale stereo SLAM is presented by Mei et al. (2009, 2010). They abandon the idea of an accurate globally referenced map in favour of constant-time operation over extended sequences. The authors argue that good *local* accuracy is sufficient for many practical applications such as path planning. They represent the environment in terms of a continuous sequence of relative locations of camera frames. An appearance-based method (Cummins & Newman, 2008) is used to detect loops and introduce additional relative links between the corresponding frames. Every landmark is represented relative to its first frame of observation. To facilitate prediction of landmark positions and association to features in the current image, the landmark is passed along the relative transformations of the graph to the current camera frame. The focus of the work is on robust and accurate local mapping. This is achieved by tracking features at multiple scales with sub-pixel accuracy. Robustness is increased by first estimating rotation between successive camera frames using a second-order image alignment method.

This rotation serves as an initial estimate of camera pose for subsequent feature matching. RANSAC is employed to robustly estimate camera pose from the established matches. Relative bundle adjustment (Sibley et al., 2009) may be used to improve the local accuracy of the map, but this is not strictly necessary.

Very recently Strasdat et al. (2010) presented a large scale monocular system. An optimisation back-end performs bundle adjustment over a sliding window of keyframes, aiming at constant-time operation during exploration. Upon loop closure, the keyframe pose graph is optimised. Similarity constraints between keyframes are used instead of rigid transformations to enable scale-drift-aware optimisation. A tracking front-end estimates the camera pose using optical-flow-based visual odometry and measurements of existing map features. New features are initialised using per-feature independent Information Filters. When a new keyframe is added, the filter estimates provide a starting point for bundle adjustment. The system achieves near real-time performance building maps of thousands of points.

2.4. Higher-Level Features and Dense Maps

The impressive recent advances in Visual SLAM primarily facilitate the localisation aspect of the problem. As described above, recent work has addressed scalable estimation methods, robust local and large-scale performance, loop closure detection, and recovery from tracking failure. However, the resulting maps are still very limited in terms of the information they provide concerning the environment. The vast majority of state-of-the-art systems build maps consisting of sparse, isolated point features. The reason is that such maps facilitate efficient localisation. Sparse point-feature systems enable robust, real-time performance by restricting image processing to a minimum.

Nevertheless, to advance the deployment of visual SLAM systems for real-world applications in robotics or augmented reality, progress on the mapping aspect of the problem is required as well. We would like to have maps that contain dense, semantically meaningful information. This is the ultimate goal that we aim for with the contributions of this thesis.

Improving maps towards dense representations is now becoming a new focus of research, as the basic tracking and localisation functionality has matured. Recent work towards dense reconstruction can be distinguished into two classes.

The first approach is to build a dense reconstruction *on top of* visual SLAM. Here, a traditional sparse visual SLAM system provides camera tracking and a point map. Dense reconstruction is treated as a separate process that uses the SLAM estimates. Typically, the inherent uncertainty is ignored, and there is no feedback of the reconstruction results into the SLAM system. Moreover, these approaches usually cannot handle the changes and refinements the SLAM estimate undergoes over time.

The second class of approaches attempt to tightly *integrate* dense reconstruction and visual SLAM. Typically, this is done by extending the SLAM map representation with higher-level features such as lines or planes. Currently, these methods lack behind the *on-top-of* class in terms of the quality and scene coverage of the reconstructed maps. However, they are attractive from a methodical point of view because they follow the core philosophy of simultaneous localisation *and* mapping. All image measurements contribute

to the solution of both problems. All information contained in the map is facilitated for localisation. The work in this thesis follows the integrative approach. In the following we discuss related work from both approaches.

2.4.1. Dense Reconstruction on Top of Visual SLAM

Dense reconstruction following the first, on-top-of-SLAM, approach has become popular very recently. This is due to the fact, that recent bundle adjustment based SLAM systems finally provide camera pose estimates that are accurate enough to allow to ignore camera pose uncertainty in the dense reconstruction stage. In fact, most of the recent work uses the Parallel Tracking And Mapping (PTAM) system by Klein & Murray (2007), a keyframe-based bundle adjustment framework. Moreover, multi-core CPUs provide enough computing power to allow dense reconstruction and SLAM to run in parallel. Highly optimised parallel GPU implementations allow to use high quality multi-view stereo approaches (Seitz et al., 2006) in near real-time now.

Stühmer et al. (2010) propose a method to estimate dense depth maps from multiple images. They adopt recent real-time high accuracy optical flow algorithms to estimate the depth map in a coarse-to-fine variational optimisation scheme on the GPU. The method is loosely integrated with PTAM in the sense that nearby keyframes are used as input to their algorithm. The sparse structure computed by PTAM is not utilised, though. Keyframe poses are treated as accurate, uncertainty is ignored for the purposes of dense reconstruction.

Newcombe & Davison (2010) present a much more complete system. They also build on top of PTAM. Like in the approach above, it is assumed that the camera motion is sufficiently accurate such that its uncertainty need not be considered in the dense reconstruction. They estimate an initial continuous scene surface from the point map provided by PTAM using scattered data interpolation techniques. The continuous surface is triangulated to provide a base mesh. A reference view and a bundle of nearby camera views is used to compute a refined partial mesh for the reference view. The base mesh is used to compute pair-wise view-predictive optical flow between the reference view and each bundle view. The base mesh is refined using constrained scene flow update based on the optical flow vectors. Multiple partial meshes are combined into an overall model of the scene. The authors present impressive results for a cluttered desktop sequence.

Also related is the work of Pan et al. (2009) who address the reconstruction of small isolated objects to acquire textured models, e.g., for computer graphic object rendering. As the object is rotated in front of a static camera a polygonal model is reconstructed. The dense modelling approach is run in parallel with a point based structure from motion framework. Similar to PTAM, the point-based reconstruction is based on bundle adjustment of keyframes. The polygonal object model is constructed from a Delaunay triangulation of the reconstructed sparse point map using a probabilistic tetrahedron carving algorithm. Reconstruction is not incremental, it is restarted from scratch each time a new keyframe is added to the point-based reconstruction.

Lovi et al. (2010) employ a tetrahedron carving algorithm to dense scene reconstruction. They present a system that is also built on top of PTAM. It uses a Delaunay triangulation of PTAM's point map to discretise 3D space. Then tetrahedra that violate

visibility constraints are carved away. The reconstruction is of lower quality than with the approaches mentioned above, in that the resulting meshes are noisy and include stray uncarved tetrahedra. However, the method has one important advantage. The reconstruction is incremental, in the sense that it changes dynamically as the estimates from the SLAM system change. In particular, the system addresses addition and removal of map points, addition and removal of visibility constraints, as well as refinement of point and camera positions. All this is handled in real-time with constant per-frame runtime.

The above methods have been demonstrated for reconstruction of close range surfaces, only. The results indicate that in these scenarios camera pose uncertainty can be ignored with no adverse effects. Generalisation to larger, e.g., outdoor, scenes remains an open issue.

A system for large scale outdoor 3D reconstruction has been presented by Pollefeys et al. (2008). Their system is specifically targeted at the reconstruction of urban scenes from drive-by video sequences. Employing GPS and inertial sensors, it is not a pure visual SLAM system. Also, loop closures are not handled and there is no global optimisation of structure or trajectory. However, their work is an impressive example of large scale dense reconstruction. Data is collected using a set of sensors mounted on a vehicle that is driving through an urban environment. The reconstruction is carried out on a computer cluster in real-time, where the algorithms are distributed across CPUs and GPUs. Camera pose estimation is provided by fusing GPS and inertial sensors, and optionally visual odometry. A sequence of depth maps is then reconstructed from the input video using multi-view plane-sweeping stereo. Adjacent depth maps are fused and finally a polygonal mesh is triangulated.

2.4.2. Dense Reconstruction Integrated with Visual SLAM

A different approach is to *integrate* dense reconstruction and visual SLAM. Work in this category aims to improve the SLAM map towards increasingly dense representations. For example, this is done by extending the map with higher-level features such as lines or planes. Maps of higher-level features are endowed with semantic information, facilitating for example interpretation by humans.

Currently, these methods do not achieve the accuracy and scene coverage of their decoupled counterparts discussed above. The big advantage however is that they are truly integrated solutions. In particular, there is no distinction between the map used for localisation and the dense reconstruction. The reconstruction is incrementally built and represented in a stochastic map. This representation maintains full correlations between map features. For example, this allows for the automatic correction of the dense reconstruction in the event of a loop closure.

Moreover, the dense information can be used directly to aid SLAM, e.g., in predicting feature visibility and appearance. For example, this is exploited by the planar feature model proposed in Chapter 7 of this thesis. These planar features can be measured directly in the images which provides increased accuracy in comparison to traditional point features.

It is to be expected that we are going to see a move towards tighter integration in the decoupled approaches as well. Ultimately, both families of methods strive for the same

goal, albeit from opposite directions. Decoupled approaches use existing dense reconstruction methods, and work towards a tighter integration with SLAM. Integrated approaches are working on extending SLAM techniques towards dense reconstruction.

Increasing Point Density

Increasing the density of point features in the map can be regarded as a very simple approach to denser mapping. Obviously, this requires to increase the size of the SLAM state representation. Systems based on a single EKF are at a disadvantage here, because their computation time grows quadratically with the map size. However, using suitable efficient feature parameterisations, such as the one proposed in Chapter 6 of this thesis, the number of features can be increased considerably.

Alternative filtering frameworks have been considered. For instance, Eade & Drummond (2006b) apply the particle filter based FastSLAM framework (Montemerlo et al., 2003) to visual SLAM. In their experiments they build a map of several hundred landmarks in real-time. A drawback of the method is the reduced consistency in comparison to EKF based systems.

A second alternative are submapping methods such as (Williams et al., 2002; Eade & Drummond, 2007; Paz et al., 2008; Piniés & Tardós, 2008). These are hierarchical approaches where a global map consists of several local maps. Usually, at the local level, maps are fully correlated. For example, Eade & Drummond (2007) employ information filtering to maintain fully correlated local maps, which form the nodes of a graph. Constraints between nodes are imposed by shared features, and are used to globally optimise the graph.

Finally, the most promising approach today is presented by sparse keyframe bundle adjustment methods such as that of Klein & Murray (2007), which handle maps comprising thousands of points.

Line Features

Edge or line features are more descriptive than point features, and maps containing line features are more accessible to human interpretation. In the image, edge features are simply characterised by a step change in intensity. This allows for robust detection under a wide range of lighting and viewpoint changes. Consequently, line features have been used for a long time in model-based visual tracking applications (e.g., Harris & Stennet, 1990; Drummond & Cipolla, 2002). However, edges are also less discriminatory than point features, which complicates data association. Moreover, measurements are possible only in one dimension, in the direction perpendicular to the edge.

Smith et al. (2006) extend the system of Davison (2003) by straight line segment features. In the EKF state, they represent a line segment by its two end-points. Image measurements are made perpendicular to the line at sample points between the projected end-points. They propose a hypothesise-and-test method for the fast detection of new line feature candidates. For this they first detect corners in the image and then perform a quick test to determine whether corner pairs are connected by lines.

Eade & Drummond (2006a) propose to use edgelets, e.g., short locally straight edge segments, as features within a particle filter based SLAM system. They argue that the use of longer edges causes problems because full edges may appear partially occluded or broken into segments, and determining their full extend may be impossible. Moreover, even curved lines in the scene can be represented by locally straight edgelets. Edgelets are represented in the filter by the 3D position of their center point and a 3D direction vector. Measurements are obtained by fitting line segments to edgels in the local neighbourhood of the predicted image projection of the edgelet.

Edgelets are also incorporated by Klein & Murray (2008) into their keyframe-based SLAM system. They argue that the inclusion of edge features into the map makes it easier to track through high-velocity motions with large amounts of motion blur. The representation of edgelets is similar to that of Eade & Drummond (2006a). The perpendicular image distances at two sample points between the projected edgelet and line detected in the image are used as measurements to update the map.

Gee & Mayol-Cuevas (2006) propose a model-based SLAM system that uses a map of line-segments. Line segments are represented by the depths of their end-points with respect to a reference camera position. Image measurements are made perpendicular to the line at sample points along the projected line segment. Their system is a combination of a fixed model tracking and independent initialisation of new model edges. Thus they can not represent correlations in the map.

Planar Features

Planar structures present another promising type of higher level structure. Locally planar structures are prevalent in man-made environments. In contrast to line features they have the potential to build truly dense environment models. Moreover, in contrast to free-form structure they can be represented by few parameters in the SLAM state vector.

Silveira et al. (2008) model the environment as a collection of planar surfaces. A planar feature is parameterised by the inverse depths of three points. They formulate visual SLAM as a nonlinear image alignment task. The plane features and the camera pose are directly estimated from image intensities using efficient second-order minimisation (Malis, 2004). A drawback of the method is the lack of a fully correlated map. Moreover, prior information is not exploited in the minimisation.

Gee et al. (2007) present an approach that builds on a point feature based visual SLAM system. Planes are detected and represented within the map. After building the point feature map, subsets of these points lying on common planes are identified. Specifically, hypotheses for plane features are generated and verified using RANSAC (Fischler & Bolles, 1981). Detected higher level planar structures are then added to the map with their own parametrisation. Point features lying on the plane can then be switched to a more compact representation with respect to the plane. Plane features are never observed directly. Instead, they are a means to group point features. Thus, they do not necessarily correspond to physical scene planes. The approach is extended by Gee et al. (2008) to line features which are composed of edgelets.

Martínez-Carranza & Calway (2009b) use planar features in an EKF based visual SLAM framework. Planes are detected using an appearance based hypothesis testing frame-

work (Martínez-Carranza & Calway, 2009a). This method, too, builds upon a point feature map to provide plane hypotheses. Detected planes are added to the map with their own parametrisation. Plane measurements are based on matching point features which are selected on-the-fly according to the current camera and plane pose.

The method is extended by Martínez-Carranza & Calway (2010) to undelayed initialisation of planar features. Every new feature that is added has the capability to become either a standard point or a plane feature. In which way a feature develops is determined by future measurements.

2.4.3. Map Representations in this Thesis

In this thesis we subscribe to the integrative approach. We propose two novel feature representations which are integrated within EKF-based visual SLAM. The inverse depth bundle parameterisation presented in Chapter 6 is a novel point feature representation. It aims at reducing the state size of a feature to increase the number of points that can be handled in real-time. This is achieved by sharing common parameters among groups of features.

We introduce a representation for planar features in Chapter 7. The presented system is the first that builds fully correlated maps using planar features that are measured directly in the images. We propose a measurement model that directly operates on image intensities. This extends and improves upon previous work in structure from motion towards robust, real-time performance.

3
Preliminaries

We assume that the reader has some familiarity with elementary geometry and linear algebra, as well as probability theory. The first part of this chapter serves to define the notation and review some useful notions. Conventions for vectors and matrices are presented in Section 3.1. Then we introduce our notation for pose transformations, i.e., transformations between 3D coordinate systems in Section 3.2. We review the quaternion and exponential representations for 3D rotations in Section 3.3. Models for the projection in monocular and stereo cameras are discussed in Section 3.5. Basic notions of probability theory are reviewed in Section 3.6.

The remainder of the chapter is concerned with introducing the probabilistic machinery that is used to tackle the visual SLAM problem. In Section 3.7 we discuss the state estimation problem (of which SLAM is an instance). The Bayes Filter, a generic recursive solution of the problem, is discussed in Section 3.8. The Kalman Filter is a practical implementation of the Bayes Filter for linear dynamical systems. It is reviewed in Section 3.9. A simple extension to nonlinear systems, the Extended Kalman Filter, is discussed in Section 3.10. Finally, in Section 3.11 we review the Iterated Extended Kalman Filter.

3.1. Vector and Matrix Notation

Vectors are denoted by bold lower-case symbols (usually lower-case), e.g., $\mathbf{x} \in \mathbb{R}^n$ is a n-vector

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}.$$

All vectors are column vectors unless otherwise noted.

Matrices are denoted by sans-serif symbols (usually upper-case), e.g., $\mathsf{A} \in \mathbb{R}^{n \times m}$ is a $n \times m$-matrix

$$\mathsf{A} = \begin{bmatrix} a_{1;1} & \dots & a_{1;m} \\ \vdots & \ddots & \vdots \\ a_{n;1} & \dots & a_{n;m} \end{bmatrix}$$

Transposition is denoted by $\top$ superscript, e.g., the transpose of the above matrix A is the $m \times n$-matrix $\mathsf{A}^\top$. The transpose of a column vector is a row vector and vice versa. Matrix inversion is denoted by a -1 superscript, e.g., $\mathsf{A}^{-1}\mathsf{A} = \mathbf{I}$.

Vectors and matrices may be partitioned into sub-vectors and sub-matrices, respectively. This is expressed using a block notation, where the elements of a vector are sub-vectors and the elements of a matrix are sub-matrices. For example, we write

$$\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ \mathbf{z} \end{pmatrix}, \quad \mathsf{A} = \begin{bmatrix} \mathsf{B} & \mathsf{C} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

to indicate that vector $\mathbf{x}$ can be partitioned into sub-vectors $\mathbf{y}$ and $\mathbf{z}$, etc. In block notation, every $\mathbf{I}$ stands for an identity matrix of appropriate size, and every $\mathbf{0}$ stands for a matrix of zeros of appropriate size. Suppose that B is $n \times m$ and C is $n \times p$ in the example above. Then we conclude that $\mathbf{I}$ is a $p \times p$ identity matrix and $\mathbf{0}$ is a $m \times p$ zero matrix.

3.2. 3D Coordinate Systems and Pose Transformations

Modeling the position and orientation of entities in 3-dimensional space requires a representation of different coordinate systems and the transformation of coordinates between them. For example, to predict the projection of a 3D point on the image plane of a camera, we have to transform the coordinates of the point from the fixed coordinate system of the environment to the coordinate system of the moving camera.

The relative orientation of two coordinate systems can be described by a rotation matrix R and a translation vector $\mathbf{t}$. Such a rotation-translation pair is referred to as a *pose*, or a *pose transformation*. The relative pose between coordinate systems $\mathcal{A}$ and $\mathcal{B}$ is the tuple

$$p^{\mathcal{BA}} = (\mathsf{R}^{\mathcal{BA}}, \mathbf{t}^{\mathcal{BA}}) \tag{3.1}$$

consisting of a rotation $\mathsf{R}^{\mathcal{BA}}$ and a translation $\mathbf{t}^{\mathcal{BA}}$. The superscripts indicate that $p^{\mathcal{BA}}$ is the pose of coordinate frame $\mathcal{A}$ measured in coordinate frame $\mathcal{B}$. The columns of the 3×3 rotation matrix $\mathsf{R}^{\mathcal{BA}}$ are unit vectors pointing in direction of the X, Y, and Z coordinate axes of frame $\mathcal{A}$ (measured in $\mathcal{B}$). The translation 3-vector $\mathbf{t}^{\mathcal{BA}}$ is the origin of frame $\mathcal{A}$ (measured in $\mathcal{B}$).

A convenient way of thinking about relative orientation is as rigid-body transformations between the coordinate systems. Taking this point of view, we can identify every rotation-translation tuple $p^{\mathcal{BA}}$ with a function $p^{\mathcal{BA}}(\cdot)$ that transforms points from frame $\mathcal{A}$ to frame $\mathcal{B}$. That is, let $\mathbf{x}^{\mathcal{A}}$ be the coordinates of point $\mathbf{x}$ expressed in frame $\mathcal{A}$. Then the coordinates $\mathbf{x}^{\mathcal{B}}$ of the same point expressed in frame $\mathcal{B}$ are obtained as

$$\mathbf{x}^{\mathcal{B}} = p^{\mathcal{BA}}(\mathbf{x}^{\mathcal{A}}) = \mathsf{R}^{\mathcal{BA}}\mathbf{x}^{\mathcal{A}} + \mathbf{t}^{\mathcal{BA}}. \tag{3.2}$$

It is well known that the set of rigid-body motions in 3-dimensional space forms a group under composition, the *special Euclidean group* $SE(3)$, (Ma et al., 2003, sect. 2.2). We will use $\circ$ to denote composition, $\cdot^{-1}$ to denote inversion, and I to denote the identity transformation.

The rotation-translation tuples resulting from pose composition and inversion are derived in the following. Let $p^{\mathcal{BA}} = (\mathsf{R}^{\mathcal{BA}}, \mathbf{t}^{\mathcal{BA}})$ and $p^{\mathcal{CB}} = (\mathsf{R}^{\mathcal{CB}}, \mathbf{t}^{\mathcal{CB}})$ be two poses. Applying their composition $p^{\mathcal{CB}} \circ p^{\mathcal{BA}}$ to a point $\mathbf{x}$ yields the same result as applying first $p^{\mathcal{BA}}$ then $p^{\mathcal{CB}}$. Using (3.2), it must hold

$$(p^{\mathcal{CB}} \circ p^{\mathcal{BA}})(\mathbf{x}) = p^{\mathcal{CB}}(p^{\mathcal{BA}}(\mathbf{x})) = \mathsf{R}^{\mathcal{CB}}\mathsf{R}^{\mathcal{BA}}\mathbf{x} + \mathsf{R}^{\mathcal{CB}}\mathbf{t}^{\mathcal{BA}} + \mathbf{t}^{\mathcal{CB}}.$$

Hence, the rotation-translation tuple for the composition $p^{\mathcal{CB}} \circ p^{\mathcal{BA}}$ is

$$p^{\mathcal{CB}} \circ p^{\mathcal{BA}} = (\mathsf{R}^{\mathcal{CB}}\mathsf{R}^{\mathcal{BA}},\ \mathsf{R}^{\mathcal{CB}}\mathbf{t}^{\mathcal{BA}} + \mathbf{t}^{\mathcal{CB}}). \tag{3.3}$$

The composition of pose $p^{\mathcal{BA}}$ and its inverse $p^{\mathcal{BA}^{-1}} = p^{\mathcal{AB}}$ must be the identity transformation, that is $p^{\mathcal{BA}} \circ p^{\mathcal{AB}} = I$. Let $(\mathsf{R}, \mathbf{t})$ be the rotation-translation associated with $p^{\mathcal{AB}}$. Noting that $(\mathbf{I}, \mathbf{0})$ represents the identity transformation, and using (3.3), it must hold

$$(\mathsf{R}^{\mathcal{BA}}\mathsf{R},\ \mathsf{R}^{\mathcal{BA}}\mathbf{t} + \mathbf{t}^{\mathcal{BA}}) = (\mathbf{I}, \mathbf{0}).$$

Hence, the rotation-translation tuple for the inverse is

$$p^{\mathcal{AB}} = p^{\mathcal{BA}^{-1}} = (\mathsf{R}^{\mathcal{BA}^{-1}},\ -\mathsf{R}^{\mathcal{BA}^{-1}}\mathbf{t}^{\mathcal{BA}}). \tag{3.4}$$

3.3. Rotation Parameterisations

In the previous section, we have represented 3-dimensional rotations by 3×3 rotation matrices. We assume that the reader is familiar with this representation. In many operations, such as the composition of transformations or the transformation of points, rotations are indeed most conveniently expressed as rotation matrices.

However, a problem with this particular representation is over-parameterisation. The rotation matrix has 9 elements while the rotation has only 3 degrees of freedom. There are six non-linear constraints on the elements of the matrix that must be enforced when performing parameter estimation or optimisation. Therefore, this representation is not well suited for these and similar tasks.

In this thesis, two rotation parametrisations are employed, namely unit quaternions and exponential coordinates. Both parameterisations have their respective advantages and disadvantages. The advantage of the exponential representation is that it is minimal: The 3 degrees of freedom of the rotation are represented by 3 parameters. Unit quaternions, however, comprise 4 parameters with one constraint which must be explicitly enforced. The advantage of the quaternion parameterisation is that it very naturally allows composition and interpolation of rotations, as well as application to points. All these operations are difficult to achieve with the exponential representation. In fact, the easiest way to, e.g., compute the composition of two exponential rotations is to convert them to quaternions or rotation matrices, carry out the operation, and convert the result back.

In the following we will introduce quaternions and exponential rotations and show how to convert between these representations.

3.3.1. Quaternions

A quaternion is an element of $\mathbb{R}^4$ and we label its components as follows.

$$\mathbf{q} = \begin{pmatrix} q_w \\ \mathbf{q_v} \end{pmatrix} = \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} \tag{3.5}$$

The set of unit length quaternions forms a group under quaternion multiplication and can be used to parameterise rotations. Consider a rotation of θ radians about the unit axis $\mathbf{v}$. The unit quaternion corresponding to this rotation is

$$\mathbf{q} = \begin{pmatrix} q_w \\ \mathbf{q_v} \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{1}{2}\theta\right) \\ \sin\left(\frac{1}{2}\theta\right)\mathbf{v} \end{pmatrix}. \tag{3.6}$$

With $\sin^2 + \cos^2 = 1$ and $\|\mathbf{v}\| = 1$ one immediately verifies that this is a unit quaternion. The identity rotation is parameterised by $\mathbf{q} = (1, 0, 0, 0)^\top$. This is a rotation by 0 radians about any axis. Obviously, the rotation axis can not be recovered from the identity quaternion.

Composition. A multiplication operation $\circ$ on quaternions is defined as

$$\mathbf{q}_1 \circ \mathbf{q}_2 = \begin{pmatrix} w_1 \\ \mathbf{v}_1 \end{pmatrix} \circ \begin{pmatrix} w_2 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2 \\ w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 \end{pmatrix}. \tag{3.7}$$

The product $\mathbf{q}_1 \circ \mathbf{q}_2$ represents the rotation obtained by first carrying out the rotation $\mathbf{q}_2$ and then carrying out the rotation $\mathbf{q}_1$. Note that quaternion multiplication is not commutative.

Inverse. The inverse $\mathbf{q}^{-1}$ of a quaternion is the rotation that reverses $\mathbf{q}$. It can be constructed by inverting the axis of rotation of $\mathbf{q}$, thus obtaining a rotation by the same angle in the opposite direction.

$$\mathbf{q}^{-1} = \begin{pmatrix} q_w \\ \mathbf{q_v} \end{pmatrix}^{-1} = \begin{pmatrix} q_w \\ -\mathbf{q_v} \end{pmatrix}. \tag{3.8}$$

Rotation of a vector. Interestingly, the rotation of a vector can be computed using only quaternion multiplication and inversion. The relationship between a point $\mathbf{x}$ and the rotated point $\mathbf{x}'$ is

$$\begin{pmatrix} 0 \\ \mathbf{x}' \end{pmatrix} = \mathbf{q} \circ \begin{pmatrix} 0 \\ \mathbf{x} \end{pmatrix} \circ \mathbf{q}^{-1}. \tag{3.9}$$

Conversion to rotation matrix. The rotation matrix corresponding to the quaternion $\mathbf{q} = (q_w, q_x, q_y, q_z)^\top$ is

$$\mathsf{R}_\mathbf{q} = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_x q_z + q_w q_y) \\ 2(q_x q_y + q_w q_z) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_y q_z + q_w q_x) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}. \tag{3.10}$$

One way to obtain this result is to apply (3.9) to rotate unit vectors pointing along the X, Y, Z coordinate axes. The rotated vectors then form the columns of the rotation matrix.

3.3.2. Exponential Coordinates

Next, we introduce the exponential rotation representation. Here, a rotation is parameterised as vectors in $\mathbb{R}^3$,

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}, \tag{3.11}$$

which we will refer to as the *exponential coordinates* of the rotation or the *exponential vector*. The rotation of θ radians about the unit axis $\mathbf{v}$ is parameterised as

$$\boldsymbol{\omega} = \theta \mathbf{v} \tag{3.12}$$

The norm $\|\boldsymbol{\omega}\|$ of the exponential vector specifies the angle, while its direction specifies the axis of rotation. The identity rotation is parameterised by $\boldsymbol{\omega} = (0, 0, 0)^\top$. Similar to the identity quaternion, this represents a rotation by 0 radians about any axis. Obviously, the rotation axis can not be recovered for the identity rotation.

We will never operate on exponential vectors directly. When we use exponential vectors to rotate points, compute the composition of two rotations or convert an exponential vector to a rotation matrix, we will always first convert them to the quaternion representation. We will discuss this conversion operation which is referred to as the exponential map shortly, as well as the inverse operation, the log map.

Conversion to rotation matrix. The rotation matrix corresponding to the exponential vector $\boldsymbol{\omega}$ will be denoted $\mathsf{R}_{\boldsymbol{\omega}}$. It is obtained by converting $\boldsymbol{\omega}$ to a quaternion $\mathbf{q} = \exp(\boldsymbol{\omega})$ (explained below) and converting the quaternion to a rotation matrix, i.e., $\mathsf{R}_{\boldsymbol{\omega}} = \mathsf{R}_{\exp(\boldsymbol{\omega})}$.

3.3.3. Exponential Map: Converting Exponential Coordinates to Quaternions

The *exponential map* $\exp(\cdot)$ transforms exponential coordinates to the corresponding quaternion representation. Using the definitions of both representations in terms of rotation axis and angle we derive

$$\exp(\boldsymbol{\omega}) = \begin{cases} (1, 0, 0, 0)^\top & \text{if } \boldsymbol{\omega} = (0, 0, 0)^\top \text{ and} \\ \begin{pmatrix} \cos\left(\frac{1}{2}\theta\right) \\ \frac{\sin\left(\frac{1}{2}\theta\right)}{\theta}\boldsymbol{\omega} \end{pmatrix} & \text{otherwise,} \end{cases} \tag{3.13}$$

where $\theta = \|\boldsymbol{\omega}\|$.

Around the identity rotation, where $\theta \to 0$, the division by θ in the last line of Equation 3.13 will cause numerical problems. However, looking at the Taylor expansion of the problematic term, Grassia (1998) points out that in fact it *is* computable and continuous at and around zero. The Taylor expansion at $\theta = 0$ of the term is

$$\frac{\sin(\frac{1}{2}\theta)}{\theta} = \frac{1}{2} - \frac{\theta^2}{48} + \frac{\theta^4}{3840} - \dots \tag{3.14}$$

We follow Grassia's solution of approximating this term in Equation 3.13 by the first two terms of it's Taylor expansion

$$\frac{\sin(\frac{1}{2}\theta)}{\theta} \approx \frac{1}{2} - \frac{\theta^2}{48} \tag{3.15}$$

in the neighbourhood of $\theta \to 0$.

3.3.4. Log Map: Converting Quaternions to Exponential Coordinates

Vice versa, the *log map* $\log(\cdot)$ converts a quaternion to exponential coordinates.

$$\log(\mathbf{q}) = \begin{cases} (0,0,0)^\top & \text{if } \mathbf{q} = (1,0,0,0)^\top \text{ and} \\ \frac{2\cos^{-1}(q_w)}{\sin(\cos^{-1}(q_w))}\mathbf{q_v} & \text{otherwise.} \end{cases} \tag{3.16}$$

Equation 3.16 can be derived as follows. From q_w in Equation 3.6 we compute the rotation angle $\theta = 2\cos^{-1}(q_w)$. Then inverting the expression for $\mathbf{q_v}$ in Equation 3.6 we obtain the unit rotation axis $\mathbf{v} = 1/\sin\left(\frac{1}{2}\theta\right) \cdot \mathbf{q_v} = 1/\sin(\cos^{-1}(q_w)) \cdot \mathbf{q_v}$. Finally, $\boldsymbol{\omega} = \theta\mathbf{v}$ gives the above formulation.

Again, the division in the last line causes problems around the identity, where we have $q_w \to 1$ and thus $\cos^{-1}(q_w) \to 0$. We can resolve this by considering the Taylor expansion. The problematic term is of the form

$$\frac{2\cos^{-1}(1-x)}{\sin(\cos^{-1}(1-x))} \quad \text{with } x = 1 - q_w. \tag{3.17}$$

At $x = 0$ the Taylor expansion is

$$\frac{2\cos^{-1}(1-x)}{\sin(\cos^{-1}(1-x))} = 2 + \frac{2x}{3} + \frac{4x^2}{15} + \frac{4x^3}{35} + \dots \tag{3.18}$$

Obviously the function is computable and continuous. To avoid numerical instability, we approximate the scaling factor by the terms up to second order of the Taylor expansion

$$\frac{2\cos^{-1}(q_w)}{\sin(\cos^{-1}(q_w))} \approx \left(2 + \frac{2(1-q_w)}{3} + \frac{4(1-q_w)^2}{15}\right)\mathbf{q_v} \tag{3.19}$$

in the neighbourhood of $q_w \to 1$.

Having defined conversion operators between exponential and quaternion representations, as well as the conversion from quaternions to rotation matrices, we are now able to freely choose parameterisations as appropriate.

3.4. Homogeneous Representation of 2D points

Consider a point on the plane, with Euclidean coordinates $\mathbf{x} = (x, y)^\top$. A *homogeneous* or *projective* representation $\tilde{\mathbf{x}}$ of the point can be constructed by appending a third coordinate $z = 1$.

$$\tilde{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

We can consider $\tilde{\mathbf{x}}$ as the coordinates of a 3D point. When it is projected to the plane $z = 1$ its image is $\mathbf{x}$. Of course, $\tilde{\mathbf{x}}$ is not the only 3D point that projects to $\mathbf{x}$. Rather, every point on the line from the origin through $\tilde{\mathbf{x}}$ projects to $\mathbf{x}$. Therefore, the homogeneous coordinates of a point are not unique. The point $\mathbf{x} = (x, y)^\top$ is represented by every

$$\tilde{\mathbf{x}} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \text{ with } x = \frac{a}{c} \text{ and } y = \frac{b}{c}. \tag{3.20}$$

One advantage of homogeneous coordinates is that 2D points at infinity can be represented by setting the third coordinate to 0.

To convert between homogeneous and Euclidean coordinates (for finite points) we define the projection h and un-projection h^{-1} functions. The projection function is defined as

$$h : \mathbb{R}^3 \to \mathbb{R}^2 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \text{ for } z \neq 0. \tag{3.21}$$

The function h computes Euclidean coordinates from homogeneous coordinates, that is, it projects its argument to the plane $z = 1$.

The un-projection function is defined as

$$h^{-1} : \mathbb{R}^2 \to \mathbb{R}^3 : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.22}$$

The function h^{-1} computes a homogeneous representation of its argument.

3.5. Perspective Projection and Camera Parameters

A camera captures light arriving from a 3-dimensional scene on an image plane, resulting in an intensity image. We model an image as a mapping from image coordinates to intensities

$$I : \mathbb{R}^2 \to \mathbb{R} \tag{3.23}$$

which is defined for the visible region of the image plane. The pixel images produced by digital cameras are discretized versions of this mapping, where both the space of image coordinates and the space of intensities are discretized.

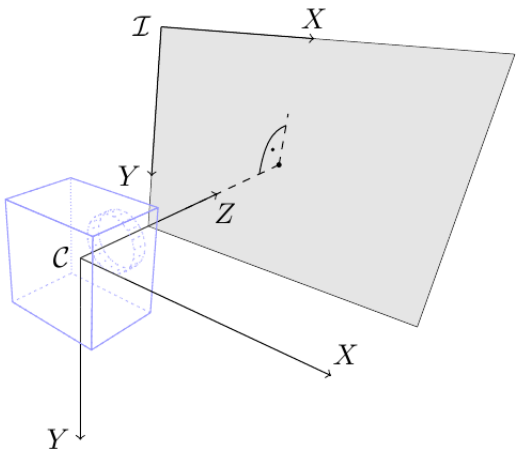


Figure 3.1.: Coordinate systems attached to the camera. The 3-dimensional camera coordinate system is $\mathcal{C}$ and the 2-dimensional image coordinate system is $\mathcal{I}$.

The mapping from 3-dimensional camera coordinates to 2-dimensional pixel coordinates on the image plane of the camera is described by the pinhole perspective projection model (Hartley & Zisserman, 2000). We assume a pure perspective projection. This means that there is no skew, i.e., the X, Y axes of the image space are perpendicular, and that there is no radial distortion.[1]

Two coordinate systems are used in the model, which are illustrated in Figure 3.1. The 3-dimensional *camera coordinate system* $\mathcal{C}$ has its origin in the projection center of the camera. The X, Y axes point in the same directions as the X, Y axes of the image coordinate system. The Z axis is perpendicular to the image plane. Camera coordinates are in units of meters.

The origin of the 2-dimensional *image coordinate system* $\mathcal{I}$ is the top left corner of the image. The X axis points to the right, the Y axis points down. That is, if we regard a discrete image as a matrix of intensity values, the X coordinate is the column and the Y axis is the row of a pixel. Camera coordinates are in units of pixel widths along the X axis and in units of pixel heights along the Y axis. Pixels need not be perfectly square, hence the pixel width and height may be different.

Monocular Projection. The projection from camera coordinates to image coordinates is a function

$$\pi : \mathbb{R}^3 \to \mathbb{R}^2 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \end{pmatrix}. \tag{3.24}$$

where f_x, f_y, u_0, v_0 are the intrinsic parameters of the camera. In particular, these parameters are

f_x, the focal length measured in units of pixel width,

[1] In practice, such distortions are removed from the camera images in a preprocessing step. This is illustrated for an example image in Figure 3.4.

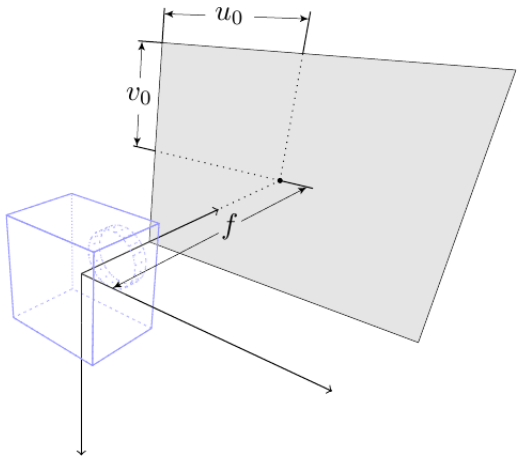


Figure 3.2.: The intrinsic camera parameters. (u_0, v_0) are the image coordinates of the principal point. f is the focal length. The focal length is given as f_x and f_y, measured in units of pixel width and height, respectively.

f_y, the focal length measured in units of pixel height,

u_0, the X image coordinate of the principal point in units of pixel width,

v_0, the Y image coordinate of the principal point in units of pixel height.

Here, the principal point is the "image center", i.e., the intersection of the Z-axis and the image plane. The focal length is the perpendicular distance from the projection center to the image plane. The intrinsic camera parameters are illustrated in Figure 3.2.

Equation 3.24 can be easily derived considering ratios in similar triangles. The ratio $\frac{z}{x}$ of a points depth z and its x-coordinate must equal the ratio $\frac{f_x}{u'}$ of the depth of the image plane f_x and the u'-coordinate of the projection. Here u' denotes the displacement in X direction from the principal point. Considering the principal point offset u_0 we end up with the formula above for $u = u' + u_0 = f_x \frac{x}{z} + u_0$. The formula for v can be derived analogously.

Camera Calibration Matrix. For a monocular camera the *camera calibration matrix* is defined as

$$\mathsf{K} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.25}$$

The K matrix is constructed from the intrinsic parameters. It allows to write the monocular projection of $\mathbf{x}$ as

$$\pi(\mathbf{x}) = h(\mathsf{K}\mathbf{x}), \tag{3.26}$$

where h is the projection function for homogeneous coordinates, Equation 3.21. It can be easily verified that this is equivalent to Equation 3.24.

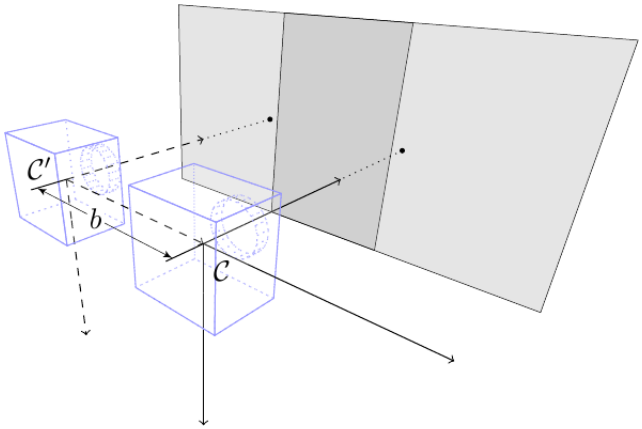


Figure 3.3.: The stereo camera model. The left and right (reference) camera are offset by the baseline b.

Normalised Image Coordinates. Consider the case $\mathsf{K} = \mathbf{I}$, that is, the camera calibration matrix is the identity matrix. This corresponds to a pure perspective projection to the plane $z = 1$. Pixel and camera units are the same length, the focal length is 1 and the image coordinate origin is at the principal point. In this case, the monocular projection is reduced to the homogeneous projection function, $\pi(\mathbf{x}) = h(\mathbf{x})$. 3D coordinates of scene points can be considered as the 2D homogeneous coordinates of image points. We will refer to the projection of a point under this identity calibration matrix as its *normalised image coordinates*.

The effect of a given calibration matrix can be removed from a projected point. Given the image projection $\mathbf{u}$ of a point in a camera with calibration matrix K, we can compute the normalised image coordinates of the point as

$$\mathbf{u}_N = h\left(\mathsf{K}^{-1} h^{-1}(\mathbf{u})\right). \tag{3.27}$$

Importantly, this is an image-to-image transformation. It is not required to know the 3D coordinates of the point.

Stereo Camera. A stereo camera consists of two monocular cameras that are displaced by the baseline b, as illustrated in Figure 3.3. We assume a *rectified* stereo setup, which means that for every scene point its projections in both images have the same Y coordinate. Specifically, we require that

- the baseline and the X axes of both cameras are parallel, and
- both cameras have the same intrinsic parameters f_x, f_y, u_0, v_0.[2]

[2] Strictly speaking, it is not necessary for the cameras of a rectified stereo setup to have identical u_0. This is assumed for simplicity here. If u_0 were different for both cameras, this could be easily corrected by shifting the image in X direction

Rectified image pairs can be obtained from any stereo configuration by warping the images to correct differences in calibration and alignment of the two cameras. One method for computing the warping parameters is described by Fusiello et al. (2000).

In the stereo case, for a 3-dimensional scene point we have a 2-dimensional projection on each image plane. We attach the reference camera coordinate system $\mathcal{C}$ to the right "eye". Then, the origin of the left camera $\mathcal{C}'$ is located at $(-b,\, 0,\, 0)^\top$ in $\mathcal{C}$. This implies that a 3D point with coordinates $\mathbf{x}_\mathcal{C} = (x,\, y,\, z)^\top$ relative to the reference camera has coordinates $\mathbf{x}_{\mathcal{C}'} = (x + b,\, y,\, z)^\top$ relative to left camera coordinate system. Thus, the point $\mathbf{x}_\mathcal{C}$ projects to

$$\mathbf{p} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \end{pmatrix} \tag{3.28}$$

$$\mathbf{p}' = \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} f_x \frac{x+b}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \end{pmatrix} \tag{3.29}$$

on the right and left image planes, respectively. The Y coordinates are identical, as indicated above. The difference between the X coordinates, $d = u' - u$, is referred to as the *disparity*. The disparity is inversely proportional to the depth of the projected point in front of the camera.

Stereo Projection. The stereo projection function

$$\pi_s : \mathbb{R}^3 \to \mathbb{R}^3 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} u \\ v \\ d \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \\ f_x \frac{b}{z} \end{pmatrix} \tag{3.30}$$

maps every 3-dimensional camera coordinate to a 3-dimensional u, v, d coordinate, where u, v is the 2-dimensional projection in the reference image and d the disparity.

Figure 3.4 illustrates the concept of disparity on an example stereo image pair. The image pair was captured using a Point Grey Bumblebee® stereo camera. This camera is also used for all practical experiments in this thesis. The figure also illustrates the rectification of the raw camera images. In the following we will assume that images have been rectified, and thus, only a pure perspective projection π (respectively π_s) relates scene and image points.

3.6. Probability Theory

In this section we review some basic definitions and rules of probability, loosely based on (MacKay, 2003). The intention is to become clear on notation and set up a point for later reference. Textbooks on probability theory commonly start out by introducing probability using discrete random variables. The concept of probability density of continuous random variables is introduced afterwards in a limiting process. In this thesis we deal with continuous random variables exclusively. Thus, for the sake of brevity, we will directly work with the continuous definitions. Lets start by defining probability density functions.

Figure 3.4.: Raw and rectified images captured with a Point Grey Bumblebee® stereo camera. The raw images in left-hand column are subject to radial distortion. The corresponding rectified images are shown in the right-hand column. Note, that radial distortion has been removed by the rectification. In the rectified images, the disparity is illustrated for a pair of corresponding points.

Definition 3.6.1 (Probability Density Function)**.** *A probability density function (pdf)* $p_{\mathbf{x}}$ *is a function* $\mathbb{R}^N \to \mathbb{R}$ *such that*

$$p_{\mathbf{x}}(\mathbf{x}) \geq 0 \text{ for all } \mathbf{x}, \text{ and}$$
$$\int p_{\mathbf{x}}(\mathbf{x})\, d\mathbf{x} = 1.$$

A N*-dimensional random vector* $\mathbf{x} \in \mathbb{R}^N$ *has the probability density function* $p_{\mathbf{x}}$ *if*

$$P(\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}) = \int_{\mathbf{a}}^{\mathbf{b}} p_{\mathbf{x}}(\mathbf{x})\, d\mathbf{x} \text{ for all } \mathbf{a}, \mathbf{b} \in \mathbb{R}^N.$$

As is common practice, we will omit the subscript in $p_{\mathbf{x}}(\mathbf{x})$ and simply write $p(\mathbf{x})$ when the meaning is clear from the context. This means that when we write $p(\mathbf{x})$ and $p(\mathbf{y})$ we are actually talking about different functions p, namely $p_{\mathbf{x}}$ and $p_{\mathbf{y}}$.

Suppose we have a probability density over a vector $\mathbf{z}$, and we partition $\mathbf{z}$ into subvectors $\mathbf{x}$ and $\mathbf{y}$. We wish to make explicit that our probability density function jointly describes the distribution of $\mathbf{x}$ and $\mathbf{y}$. We can use the following formal notation.

Definition 3.6.2 (Joint Probability Density)**.** *The joint probability density of vectors* $\mathbf{x} \in \mathbb{R}^N$ *and* $\mathbf{y} \in \mathbb{R}^M$ *will be denoted* $p_{\mathbf{xy}}(\mathbf{x}, \mathbf{y})$*, where* $p_{\mathbf{xy}}$ *is a probability density function according to Definition 3.6.1 over* $\mathbb{R}^N \times \mathbb{R}^M$*.*

For joint densities, too, we will usually shortly write $p(\mathbf{x}, \mathbf{y})$ instead of $p_{\mathbf{xy}}(\mathbf{x}, \mathbf{y})$.

Suppose that pairs of random values $(\mathbf{x}, \mathbf{y})$ are drawn according to the joint probability density $p(\mathbf{x}, \mathbf{y})$. Further suppose that we are only interested in the $\mathbf{x}$ component of every pair. Specifically, we wish to know the distribution of $\mathbf{x}$ according to the joint density, regardless of the values of $\mathbf{y}$. This can be achieved by integrating the joint probability over all possible values of $\mathbf{y}$. The resulting distribution $p(\mathbf{x})$ is referred to as the marginal distribution because it is obtained by marginalising, or integrating out, the other variables from the joint.

Proposition 3.6.3 (Marginal Probability Density)**.** *Given the joint probability density* $p(\mathbf{x}, \mathbf{y})$*, the marginal probability density of* $\mathbf{x}$ *is*

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{y}) d\mathbf{y}.$$

Suppose we already know the value of $\mathbf{y}$ and are interested in the probability of $\mathbf{x}$ conditioned on this fact. For example, suppose that pairs of random values $(\mathbf{x}, \mathbf{y})$ are drawn according to the joint probability density $p(\mathbf{x}, \mathbf{y})$. Now we take only those pairs where $\mathbf{y}$ has a specific value and consider the distribution of $\mathbf{x}$ among those pairs. This conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ is denoted $p(\mathbf{x} \,|\, \mathbf{y})$ and is defined as follows.

Definition 3.6.4 (Conditional Probability Density)**.** *Given the joint probability density* $p(\mathbf{x}, \mathbf{y})$ *and the marginal probability density* $p(\mathbf{y})$*, the conditional probability density of* $\mathbf{x}$ *given* $\mathbf{y}$ *is defined as*

$$p(\mathbf{x} \,|\, \mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}$$

for $p(\mathbf{y}) > 0$*.*

Now we can derive the two fundamental rules of probability theory, the product and sum rules.

Proposition 3.6.5 (Product Rule)**.**

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \,|\, \mathbf{y})\; p(\mathbf{y})$$

The product rule follows directly from Definition 3.6.4. It is also sometimes called the *chain rule*.

Proposition 3.6.6 (Sum Rule)**.**

$$p(\mathbf{x}) = \int p(\mathbf{x} \,|\, \mathbf{y})\; p(\mathbf{y})\; d\mathbf{y}.$$

The sum rule follows directly from Proposition 3.6.3 and Proposition 3.6.5. It is also called the *theorem of total probability*.

From Proposition 3.6.5 we have $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \,|\, \mathbf{y})\; p(\mathbf{y}) = p(\mathbf{y} \,|\, \mathbf{x})\; p(\mathbf{x})$. Rearranging this equation gives

Proposition 3.6.7 (Bayes' Theorem)**.**

$$p(\mathbf{x} \,|\, \mathbf{y}) = \frac{p(\mathbf{y} \,|\, \mathbf{x})\; p(\mathbf{x})}{p(\mathbf{y})} = \frac{p(\mathbf{y} \,|\, \mathbf{x})\; p(\mathbf{x})}{\int p(\mathbf{y} \,|\, \mathbf{x})\; p(\mathbf{x})\; d\mathbf{x}}.$$

In the final expression, we have applied the sum rule to write the denominator as $p(\mathbf{y}) = \int p(\mathbf{y} \,|\, \mathbf{x})\; p(\mathbf{x})\; d\mathbf{x}$. Note that here the denominator is calculated by integrating the numerator over all values of $\mathbf{x}$. For a given $\mathbf{y}$, the denominator can be regarded as a normalisation constant that ensures that $p(\mathbf{x} \,|\, \mathbf{y})$ is a proper probability density function, i.e., integrates to 1. Hence, Bayes' rule is often written as

$$p(\mathbf{x} \,|\, \mathbf{y}) = \eta\; p(\mathbf{y} \,|\, \mathbf{x})\; p(\mathbf{x}) \tag{3.31}$$

where η denotes the appropriate normalisation constant.

In practice, Bayes rule is often used to update uncertain knowledge with new information. In such a scenario, $p(\mathbf{x})$ expresses the *a priori* knowledge about a state $\mathbf{x}$. Observations $\mathbf{y}$ provide new information about the state $\mathbf{x}$, and we have a model of how the certain states *cause* certain observations. This model is described by the conditional probability $p(\mathbf{y} \,|\, \mathbf{x})$. Now assume that we learn the actual value $\mathbf{y}$, i.e., we make an observation. Then Bayes' rule gives us a method to update our knowledge with this new information. That is, Bayes' rule allows to calculate the *a posteriori* probability $p(\mathbf{x} \,|\, \mathbf{y})$ conditioned on the fact that we now know the value of $\mathbf{y}$.

Please note, that the rules established in Propositions 3.6.5–3.6.7 can be applied conditioned on arbitrary background knowledge, i.e., conditioned on additional random variables. For example, conditioning the product rule on the random variable $\mathbf{z}$ we have

$$p(\mathbf{x}, \mathbf{y} \,|\, \mathbf{z}) = p(\mathbf{x} \,|\, \mathbf{y}, \mathbf{z})\; (\mathbf{y} \,|\, \mathbf{z}). \tag{3.32}$$

We will further need the notions of independence and conditional independence. Two random variable are independent if they carry no information about each other. If $\mathbf{x}$ and $\mathbf{y}$ are independent, then learning the exact value of $\mathbf{x}$ will tell us nothing about the value of $\mathbf{y}$, and vice versa. Formally, independence is defined as follows.

Definition 3.6.8 (Independence)**.** *Two random variables* $\mathbf{x}$ *and* $\mathbf{y}$ *are independent if and only if*

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})\, p(\mathbf{y})$$

Independence can also be conditioned on background knowledge:

Definition 3.6.9 (Conditional Independence)**.** *Two random variables* $\mathbf{x}$ *and* $\mathbf{y}$ *are conditionally independent given* $\mathbf{z}$ *if and only if*

$$p(\mathbf{x}, \mathbf{y} \,|\, \mathbf{z}) = p(\mathbf{x} \,|\, \mathbf{z})\, p(\mathbf{y} \,|\, \mathbf{z})$$

Conditional independence expresses that, given that the value of $\mathbf{z}$ is known, learning the value of variable $\mathbf{y}$ provides no additional information about $\mathbf{x}$. Using Proposition 3.6.5 conditioned on $\mathbf{z}$, the conditional independence of $\mathbf{x}$ and $\mathbf{y}$ can be equivalently expressed as

$$p(\mathbf{x} \,|\, \mathbf{y}, \mathbf{z}) = p(\mathbf{x} \,|\, \mathbf{z}) \tag{3.33}$$

or

$$p(\mathbf{y} \,|\, \mathbf{x}, \mathbf{z}) = p(\mathbf{y} \,|\, \mathbf{z}). \tag{3.34}$$

An important probability density function in practice is the Gaussian Normal distribution. It is a parametric distribution that is fully characterised by a mean vector and covariance matrix. Its special properties and wide applicability make it a central tool in probabilistic inference.Throughout this thesis, uncertain knowledge will be modeled using Gaussian distributions.

Definition 3.6.10 (Multivariate Gaussian Normal distribution)**.** *Let* $\boldsymbol{\mu} \in \mathbb{R}^N$ *be a vector and* $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$ *be a symmetric positive semidefinite matrix. The* N*-dimensional Gaussian probability density function with mean* $\boldsymbol{\mu}$ *and covariance* $\boldsymbol{\Sigma}$ *is*

$$p(\mathbf{x}) = \frac{1}{|2\pi\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\,(\mathbf{x} - \boldsymbol{\mu})^\top\, \boldsymbol{\Sigma}^{-1}\,(\mathbf{x} - \boldsymbol{\mu})\right).$$

To state that a random vector $\mathbf{x} \in \mathbb{R}^N$ *has a Gaussian probability density function with mean* $\boldsymbol{\mu}$ *and covariance* $\boldsymbol{\Sigma}$ *we write*

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

The probability density for a specific value $\mathbf{x}$ *is denoted as*

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}\,;\, \boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

Gaussian distributions have many nice properties. If a joint distribution $p(\mathbf{x}, \mathbf{y})$ is Gaussian then the marginal distributions $p(\mathbf{x})$ and $p(\mathbf{y})$, as well as the conditional distributions $p(\mathbf{x} \,|\, \mathbf{y})$ and $p(\mathbf{y} \,|\, \mathbf{x})$ are Gaussians, too. If a variable $\mathbf{y}$ is related to a Gaussian random variable $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ by an affine function $\mathbf{y} = \mathsf{A}\mathbf{x} + \mathbf{b}$ then $\mathbf{y}$ is also Gaussian distributed with $\mathbf{y} \sim \mathcal{N}(\mathsf{A}\boldsymbol{\mu} + \mathbf{b}, \mathsf{A}\boldsymbol{\Sigma}\mathsf{A}^\top)$.

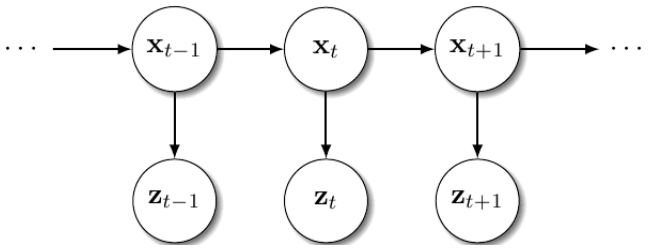


Figure 3.5.: Hidden Markov model representing a dynamical system without control inputs.

3.7. State Estimation in Dynamical Systems

Consider a dynamical system like the one depicted in Figure 3.5. The state of the system at time t is denoted by $\mathbf{x}_t$ where $\mathbf{x}_t$ takes a value from a continuous state space. Time evolves in discrete steps $t = 0, 1, 2, \ldots$

The state is not observable directly, but at every time step $t = 1, 2, \ldots$ we make an observation or measurement $\mathbf{z}_t$ of some aspects of the system. There is a functional relationship between $\mathbf{x}_t$ and $\mathbf{z}_t$. This relationship need not be deterministic in general. It is described by a stochastic *measurement model* $p(\mathbf{z}_t \,|\, \mathbf{x}_t)$. The assumption is made, that the measurement at time t only directly depends on state at time t. If the exact value $\mathbf{x}_t$ is known, then learning the values of (all or some) other states or measurements provides no additional information on $\mathbf{z}_t$. In particular, if $\mathbf{x}_t$ is known, then measurements made in the past provide no additional information on $\mathbf{z}_t$. This is expressed in the following conditional independence relation.

$$p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{z}_{1:t-1}) = p(\mathbf{z}_t \,|\, \mathbf{x}_t) \tag{3.35}$$

In the equation above and in the following we use the convention to denote a sequence $\mathbf{z}_i, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_j$ by $\mathbf{z}_{i:j}$

The evolution of the system is modeled as a discrete-time Markov process. The transition probability from state $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ is described by a stochastic *process model* $p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$. We make a conditional independence assumption for the process model as well. Given the exact value of $\mathbf{x}_{t-1}$, then learning the values of (all or some) other states or measurements *that precede* $\mathbf{x}_t$ provides no additional information on $\mathbf{x}_t$. In particular,

$$p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}) = p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \tag{3.36}$$

Usually, the dynamical system is described with additional control inputs $\mathbf{u}_t$ to the process model. The visual SLAM problem considered in this thesis has no such inputs. Thus, we have omitted control inputs to simplify the presentation.

The conditional independence assumptions (3.35) and (3.36) are called Markov assumptions. Figure 3.5 also illustrates the conditional dependence (respective independence) structure of the problem: Two nodes are conditionally independent given a set of nodes that blocks all paths between the two nodes. For example, $\mathbf{x}_t$ and $\mathbf{z}_{t-1}$ are conditionally

independent given $\mathbf{x}_{t-1}$. If $\mathbf{x}_{t-1}$ is known, then learning the value of $\mathbf{z}_{t-1}$ provides no additional information about $\mathbf{x}_t$.

The problem we wish to solve now is, at every time step t, to compute state $\mathbf{x}_t$ taking into account all measurements available at this time, i.e., $\mathbf{z}_{1:t}$. Because the state is not directly observable and measurements are corrupted by noise, in general it is impossible to obtain the exact value of $\mathbf{x}_t$. The correct way to represent our beliefs about $\mathbf{x}_t$ then, is by assigning a probability to every possible state, i.e., we are interested in the probability density $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$. Formally, we state the problem as follows.

Definition 3.7.1 (State Estimation Problem)**.** *Given a dynamical system with process model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and measurement model $p(\mathbf{z}_t \mid \mathbf{x}_t)$, let the initial state of the system be distributed with $p(\mathbf{x}_0)$. Let $\mathbf{z}_{1:n}$ be the sequence of measurements up to the current time step n. The state estimation problem consists in calculating $p(\mathbf{x}_n \mid \mathbf{z}_{1:n})$ which describes the knowledge about the current system state.*

Two characteristics are important to understand here. First, note that $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$ is to be understood as a function of $\mathbf{x}_t$, assigning a probability density to every concrete value of $\mathbf{x}_t$. We are interested in the whole function, not just it's value at a specific point in the state space.

Second, there is an emphasis on causality. At every time t we want the best estimate we can get with the information available *now*. Although future measurements $\mathbf{z}_{t+1:\dots}$ provide additional information about $\mathbf{x}_t$, they are not available at time t. At any given time t, the state $\mathbf{x}_t$ must be estimated using only the observations made up to this point.

This meets well the requirements for the SLAM problem where the focus is also on providing real-time estimates of a system. Indeed, visual SLAM can be described as a state estimation problem: We have a hidden state that we want to estimate from observations. The state at any given time t consists of the pose of the camera and a parametric description of the world, i.e., the map. The observations we make of this state are the projections of the world in the camera images. If the camera is integrated in a feedback loop, e.g., controlling a mobile robot, we require that estimates of the system state are available on the fly.

3.8. The Bayes Filter

A general, recursive solution to the state estimation problem, referred to as the *Bayes Filter*, will be described in this section along the lines of Thrun et al. (2005).

The Bayes Filter is given in Algorithm 1. It maintains a *belief* distribution $\text{bel}(\mathbf{x}_t)$ which at every time step t describes the current knowledge about the state. The belief is updated recursively by alternating a *prediction* step and an *update* step.

In the prediction step, the prior belief $\overline{\text{bel}}(\mathbf{x}_t)$ is computed from the belief of the previous time step $\text{bel}(\mathbf{x}_{t-1})$. The prior belief describes the *a priori* knowledge about the current state, without having seen the current measurement yet. The prediction step is given in line 4 of the algorithm. The calculation involves the process model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. It is multiplied by the belief distribution about the previous state $\mathbf{x}_{t-1}$ to give a joint distribution about the current and previous state. Then, the previous $\mathbf{x}_{t-1}$ is integrated out from the joint distribution to obtain the prior belief about the current state.

Algorithm 1: BayesFilter

Input: Initial state distribution $p(\mathbf{x}_0)$, Sequence of measurements $\mathbf{z}_{1:n}$
Output: State distribution $p(\mathbf{x}_n \mid \mathbf{z}_{1:n})$

1 $\text{bel}(\mathbf{x}_0) := p(\mathbf{x}_0)$
2 $t := 1$
3 **while** $t \leq n$ **do**
4 $\quad \overline{\text{bel}}(\mathbf{x}_t) := \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \, \text{bel}(\mathbf{x}_{t-1}) \, d\mathbf{x}_{t-1}$ `// prediction step`
5 $\quad \text{bel}(\mathbf{x}_t) := \eta \, p(\mathbf{z}_t \mid \mathbf{x}_t) \, \overline{\text{bel}}(\mathbf{x}_t)$ `// update step`
6 $\quad t := t + 1$
7 **end**
8 **return** $\text{bel}(\mathbf{x}_n)$

In the update step the prior belief is updated with the current measurement $\mathbf{z}_t$ to give the posterior belief $\text{bel}(\mathbf{x}_t)$. The posterior belief describes the *a posteriori* knowledge about the current state given all measurements up to the current one. The update step is given in line 5 of the algorithm. The calculation uses the generative measurement model $p(\mathbf{z}_t \mid \mathbf{x}_t)$ to integrate the current measurement $\mathbf{z}_t$. The update step is an application of Bayes' theorem. The factor η is a normalisation constant that ensures that $\text{bel}(\mathbf{x}_t)$ is a true probability density function, i.e., over all $\mathbf{x}_t$ it integrates to 1.

The Bayes Filter is not a practical algorithm that can be implemented for arbitrary stochastic models and belief distributions. Note that $\overline{\text{bel}}(\mathbf{x}_t)$ and $\text{bel}(\mathbf{x}_t)$ refer to the full probability density functions. That is, the calculations in lines 4 and 5 must be carried out for all possible values of $\mathbf{x}_t$. We need to compute the full distributions. This can only be practically implemented for special problems that are either defined over a discrete state space or have special belief and model characteristics that allow to solve the calculations in closed form. One such instance will be discussed in the next section, namely linear dynamic systems which can be solved in closed form using the Kalman Filter.

In the following, we show that the Bayes Filter is correct, i.e., we show that the recursively computed belief indeed represents the solution to the state estimation problem. Specifically, we show by induction that for all t

$$\text{bel}(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{z}_{1:t}). \tag{3.37}$$

In the base case $t = 0$, the statement holds trivially because we set $\text{bel}(\mathbf{x}_0) = p(\mathbf{x}_0)$ in line 1 of the algorithm.

For the inductive step, assume that the statement holds for $t - 1$, i.e.,

$$\text{bel}(\mathbf{x}_{t-1}) = p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}). \tag{3.38}$$

We first proof the correctness of the prediction step. We show that the computed prior belief indeed describes the *a priori* knowledge about the state at time t, i.e., we show that

$\overline{\text{bel}}(\mathbf{x}_t) = p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1})$. In line 4, the prior belief is computed as

$$\overline{\text{bel}}(\mathbf{x}_t) = \int p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \;\text{bel}(\mathbf{x}_{t-1}) \; d\mathbf{x}_{t-1} \tag{3.39}$$

We substitute $\text{bel}(\mathbf{x}_{t-1})$ using the induction hypothesis (3.38) and obtain

$$\overline{\text{bel}}(\mathbf{x}_t) = \int p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \; p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}) \; d\mathbf{x}_{t-1} \tag{3.40}$$

Using the Markov assumption (3.36), we can substitute $p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1})$. We obtain

$$\overline{\text{bel}}(\mathbf{x}_t) = \int p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}) \; p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}) \; d\mathbf{x}_{t-1}. \tag{3.41}$$

Using the sum rule, Proposition 3.6.6, conditioned on $\mathbf{z}_{1:t-1}$ we obtain

$$\overline{\text{bel}}(\mathbf{x}_t) = p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1}), \tag{3.42}$$

which shows the correctness of the prediction step.

It remains to proof the update step. That is, we show that the computed posterior belief indeed describes the *a posteriori* knowledge about the state at time t after having seen the measurement $\mathbf{z}_t$. In line 5 of the algorithm, the posterior belief is computed as

$$\text{bel}(\mathbf{x}_t) = \eta \; p(\mathbf{z}_t \,|\, \mathbf{x}_t) \; \overline{\text{bel}}(\mathbf{x}_t). \tag{3.43}$$

Here, η denotes a normalisation factor such that $\text{bel}(\mathbf{x}_t)$ integrates to 1. Substituting the prior belief according to (3.42) we obtain

$$\text{bel}(\mathbf{x}_t) = \eta \; p(\mathbf{z}_t \,|\, \mathbf{x}_t) \; p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1}). \tag{3.44}$$

Using the Markov assumption (3.35) we can substitute $p(\mathbf{z}_t \,|\, \mathbf{x}_t) = p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{z}_{1:t-1})$. We obtain

$$\text{bel}(\mathbf{x}_t) = \eta \; p(\mathbf{z}_t \,|\, \mathbf{x}_t, \mathbf{z}_{1:t-1}) \; p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1}). \tag{3.45}$$

Using Bayes' rule, Proposition 3.6.7, conditioned on $\mathbf{z}_{1:t-1}$ we obtain

$$\text{bel}(\mathbf{x}_t) = p(\mathbf{x}_t \,|\, \mathbf{z}_t, \mathbf{z}_{1:t-1}) = p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t}). \tag{3.46}$$

This shows that (3.37) holds for t and completes the inductive step. The correctness of the Bayes Filter follows by induction.

3.9. The Kalman Filter

The Kalman Filter is a set of equations that exactly implement the Bayes Filter prediction and update steps for *linear Gaussian systems*. In fact, such systems are also often referred to as *Kalman Filter models* (Roweis & Ghahramani, 1999). A linear Gaussian systems is characterised by process and measurement models that can be described as linear functions with additive Gaussian noise. More precisely, the system satisfies the following conditions:

1. $\mathbf{x}_t \in \mathbb{R}^{n_t}$ and $\mathbf{z}_t \in \mathbb{R}^{m_t}$ for some n_t, m_t, for all t. That is, states and measurements are vectors of reals.

2. The process model (evolution of the system state) is described by the state transition function
$$\mathbf{x}_t = \mathsf{A}_t \mathbf{x}_{t-1} + \boldsymbol{\varepsilon}_t. \tag{3.47}$$
Here, A_t is a $n_t \times n_{t-1}$ matrix describing the noise-free model. The vector $\boldsymbol{\varepsilon}_t \in \mathbb{R}^{n_t}$ is the *process noise*. The process noise is drawn from a white, zero-mean Gaussian distribution with covariance Q_t, i.e., $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathsf{Q}_t)$.

3. The measurement model (generation of observations) is described by the measurement function
$$\mathbf{z}_t = \mathsf{H}_t \mathbf{x}_t + \boldsymbol{\delta}_t. \tag{3.48}$$
Here, H_t is a $m_t \times n_t$ matrix describing the noise-free measurement. The measurement noise is drawn from a white, zero-mean Gaussian distribution with covariance R_t, i.e., $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathsf{R}_t)$.

Note that the matrices describing the process and measurement models, A_t and H_t, as well as the dimension of the state and measurement vectors might change with each time step.

Equations (3.47) and (3.48) allow us to write the conditional probability densities for the Bayes Filter process and measurement models as Gaussians,

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t\,;\, \mathsf{A}_t\mathbf{x}_{t-1}, \mathsf{Q}_t) \tag{3.49}$$
$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{z}_t\,;\, \mathsf{H}_t\mathbf{x}_t, \mathsf{R}_t). \tag{3.50}$$

Assuming that the initial belief about the state is Gaussian distributed, i.e., $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ for some $\boldsymbol{\mu}_0$, $\boldsymbol{\Sigma}_0$, this implies that all future beliefs and measurements will also be Gaussian distributed. This means that at every time step the prior and posterior belief distributions can be specified by their respective mean vectors and covariance matrices, i.e.,

$$\overline{\mathrm{bel}}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t\,;\, \overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\Sigma}}_t) \tag{3.51}$$
$$\mathrm{bel}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t\,;\, \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t). \tag{3.52}$$

The Kalman Filter is a set of equations to update the mean and covariance matrices describing the belief. The Kalman Filter equations can be obtained by substituting (3.49) and (3.50) into the Bayes Filter equations. For a detailed derivation we refer to (Thrun et al., 2005, section 3.2.4).

The Kalman Filter is presented in Algorithm 2. For linear Gaussian systems, it is an exact implementation of the Bayes Filter. The prior and posterior beliefs are represented by their respective mean vectors and covariance matrices. The Kalman Filter follows the same schema of alternating prediction and update for each time step. The implementation of the PREDICT and UPDATE functions is detailed in Algorithms 3 and 4, respectively.

In the prediction step, the prior belief is computed by passing the previous state distribution through the process model. Intuitively, we predict what happens in the period of

Algorithm 2: KalmanFilter

Input: Initial belief distribution $\boldsymbol{\mu}_0$, $\boldsymbol{\Sigma}_0$, Sequence of measurements $\mathbf{z}_{1:n}$
Output: Belief distribution $\boldsymbol{\mu}_n$, $\boldsymbol{\Sigma}_n$

```
1 t := 1
2 while t ≤ n do
3 |   μ̄_t, Σ̄_t := Predict(μ_{t−1}, Σ_{t−1}, A_t, Q_t)
4 |   μ_t, Σ_t := Update(μ̄_t, Σ̄_t, H_t, R_t, z_t)
5 |   t := t + 1
6 end
7 return μ_n, Σ_n
```

Algorithm 3: Predict

Input: Belief distribution $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, Process model A, Q
Output: Prior belief distribution $\overline{\boldsymbol{\mu}}$, $\overline{\boldsymbol{\Sigma}}$

```
1 μ̄ := Aμ
2 Σ̄ := AΣA^⊤ + Q
3 return μ̄, Σ̄
```

"blindness" between measurements, which normally means that our uncertainty about the state increases. We can observe this increase in line 2 of the Algorithm 3, which computes the prior covariance. Here, the covariance is projected through the process model A and then uncertainty is increased by adding the process noise covariance Q.

The update calculations are given in Algorithm 4. The prior estimate is refined with new information from the current measurement $\mathbf{z}$. The mean and covariance of the resulting posterior distribution are computed in lines 4 and 5. The new information from the measurement is represented by the *innovation* $\boldsymbol{\nu}$ and *innovation covariance* S. It is weighted with the *Kalman gain* K which trades off how much we trust in the prediction and the measurement, respectively.

The innovation $\boldsymbol{\nu}$ is computed in line 1 of Algorithm 4. The innovation is the difference between actual measurement and the predicted measurement. Here, the predicted measurement $\mathsf{H}\overline{\boldsymbol{\mu}}$ is the measurement we would make if our prior estimate was correct and there was no measurement noise. The innovation covariance S is computed in line 2. It is the sum of the prior state covariance projected into the measurement space and the measurement noise covariance.

The innovation describes the deviation between the actual measurement and the predicted measurement. Let us assume that we do not know the actual value of the measurement $\mathbf{z}$ yet. Then our belief about the innovation is described by a zero-mean Gaussian distributed with covariance S.

$$p(\boldsymbol{\nu} \mid \mathbf{z}_{1:t-1}) = \mathcal{N}(\boldsymbol{\nu}\,;\, \mathbf{0}, \mathsf{S}). \tag{3.53}$$

Algorithm 4: UPDATE

Input: Prior belief distribution $\overline{\boldsymbol{\mu}}$, $\overline{\boldsymbol{\Sigma}}$, Measurement model H, R, Measurement $\mathbf{z}$
Output: Posterior belief distribution $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

$\boldsymbol{\nu} := \mathbf{z} - \mathsf{H}\overline{\boldsymbol{\mu}}$
$\mathsf{S} := \mathsf{H}\overline{\boldsymbol{\Sigma}}\mathsf{H}^\top + \mathsf{R}$
$\mathsf{K} := \overline{\boldsymbol{\Sigma}}\mathsf{H}^\top\mathsf{S}^{-1}$
$\boldsymbol{\mu} := \overline{\boldsymbol{\mu}} + \mathsf{K}\boldsymbol{\nu}$
$\boldsymbol{\Sigma} := \overline{\boldsymbol{\Sigma}} - \mathsf{K}\mathsf{S}\mathsf{K}^\top$
return $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

Equivalently, our belief about the measurement is Gaussian distributed with

$$p(\mathbf{z}_t \,|\, \mathbf{z}_{1:t-1}) = \mathcal{N}(\mathbf{z}_t \,;\, \mathsf{H}\overline{\boldsymbol{\mu}}_t, \mathsf{S}). \tag{3.54}$$

Thus, the innovation distribution describes the predicted measurement and its uncertainty. This is of importance in the visual SLAM scenario, because it allows to focus low-level image processing resources to image areas where the measurement is expected with high probability. This approach is referred to as active search and will be discussed in Section 4.7.

3.10. The Extended Kalman Filter

The Kalman filter is very appealing because it is both computationally efficient and provably optimal for linear Gaussian systems. However, its assumptions are rarely met in practice. More often than not, practical state estimation problems have nonlinear process and measurement models. The Extended Kalman Filter (EKF) is a simple approximation to cope with such systems. Its basic idea is to replace the nonlinear functions by linear approximations using first order Taylor expansion. This linearisation is performed around the current mean of state estimate, in every time-step. Then the Kalman filter equations are applied to the linearised models.

We characterize a dynamical system by process and measurement models that can be described as possibly nonlinear functions of the system state and Gaussian noise. More precisely, the system satisfies the following conditions:

1. $\mathbf{x}_t \in \mathbb{R}^{n_t}$ and $\mathbf{z}_t \in \mathbb{R}^{m_t}$ for some n_t, m_t, for all t.

2. The process model is described by the state transition function

$$\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t) \tag{3.55}$$

where $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathsf{Q}_t)$ is white, zero-mean Gaussian process noise.

3. The measurement model is described by the measurement function

$$\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t, \boldsymbol{\delta}_t) \tag{3.56}$$

where $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}, \mathsf{R}_t)$ is white, zero-mean Gaussian measurement noise.

Note that the functions describing the process and measurement models, $\mathbf{f}_t$ and $\mathbf{h}_t$, as well as the dimension of the state and measurement vectors might change with each time step. Moreover, observe that the assumption of Gaussian noise here is not a strong restriction: A given continuos noise distribution can be approximated as a sum of Gaussians. Thus, any noise model can be approximated by making the dimension of the noise vectors large enough and encoding the summation in the functions $\mathbf{f}_t$ and $\mathbf{h}_t$, respectively.

Similar to the Kalman Filter, the EKF models the prior and posterior belief distributions as Gaussians. At every time step they are respectively represented by a mean vector and covariance matrix. For the belief calculations of the EKF, the nonlinear process and measurement models are approximated by linearisations. The linearisation is performed in every time step around the mean of the belief distribution.

Let us consider the process model (3.55). A first order Taylor expansion yields

$$\begin{aligned}\mathbf{x}_t &= \mathbf{f}_t(\mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t) \\ &\approx \mathbf{f}_t(\boldsymbol{\mu}_{t-1}, \mathbf{0}) + \left.\frac{\partial \mathbf{f}_t}{\partial \mathbf{x}_{t-1}}\right|_{(\boldsymbol{\mu}_{t-1}, \mathbf{0})} (\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) + \left.\frac{\partial \mathbf{f}_t}{\partial \boldsymbol{\varepsilon}_t}\right|_{(\boldsymbol{\mu}_{t-1}, \mathbf{0})} \boldsymbol{\varepsilon}_t,\end{aligned} \tag{3.57}$$

where the expansion is performed at the expected values of the belief of the previous time-step, $E[\mathbf{x}_{t-1}] = \boldsymbol{\mu}_{t-1}$, and the process noise, $E[\boldsymbol{\varepsilon}_t] = \mathbf{0}$. The above approximation yields a linear process model similar to the form used in the Kalman Filter. It allows to approximate the conditional probability density for the Bayes Filter process model as Gaussian,

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}) \approx \mathcal{N}(\mathbf{x}_t\,;\, \mathbf{f}_t(\boldsymbol{\mu}_{t-1}, \mathbf{0}) + \frac{\partial \mathbf{f}_t}{\partial \mathbf{x}_{t-1}}(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}),\, \frac{\partial \mathbf{f}_t}{\partial \boldsymbol{\varepsilon}_t}\, \mathsf{Q}_t\, \frac{\partial \mathbf{f}_t}{\partial \boldsymbol{\varepsilon}_t}^\top). \tag{3.58}$$

Similarly, a first order Taylor expansion of the measurement model (3.56) yields

$$\begin{aligned}\mathbf{z}_t &= \mathbf{h}_t(\mathbf{x}_t, \boldsymbol{\delta}_t) \\ &\approx \mathbf{h}_t(\overline{\boldsymbol{\mu}}_t, \mathbf{0}) + \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t}\right|_{(\overline{\boldsymbol{\mu}}_t, \mathbf{0})} (\mathbf{x}_t - \overline{\boldsymbol{\mu}}_t) + \left.\frac{\partial \mathbf{h}_t}{\partial \boldsymbol{\delta}_t}\right|_{(\overline{\boldsymbol{\mu}}_t, \mathbf{0})} \boldsymbol{\delta}_t,\end{aligned} \tag{3.59}$$

where the expansion is performed at the expected values of the prior belief, $E[\mathbf{x}_t] = \overline{\boldsymbol{\mu}}_t$, and the measurement noise, $E[\boldsymbol{\delta}_t] = \mathbf{0}$. This approximation yields a linear measurement model similar to the form used in the Kalman Filter. It allows to approximate the conditional probability density for the Bayes Filter measurement model as Gaussian,

$$p(\mathbf{z}_t|\mathbf{x}_t) \approx \mathcal{N}(\mathbf{z}_t\,;\, \mathbf{h}_t(\overline{\boldsymbol{\mu}}_t, \mathbf{0}) + \frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t}(\mathbf{x}_t - \overline{\boldsymbol{\mu}}_t),\, \frac{\partial \mathbf{h}_t}{\partial \boldsymbol{\delta}_t}\, \mathsf{R}_t\, \frac{\partial \mathbf{h}_t}{\partial \boldsymbol{\delta}_t}^\top). \tag{3.60}$$

Algorithm 5: ExtendedKalmanFilter

Input: Initial belief distribution $\boldsymbol{\mu}_0$, $\boldsymbol{\Sigma}_0$, Sequence of measurements $\mathbf{z}_{1:n}$
Output: Belief distribution $\boldsymbol{\mu}_n$, $\boldsymbol{\Sigma}_n$

$t := 1$
while $t \leq n$ **do**
  $\overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\Sigma}}_t := \text{PredictEkf}\left(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \mathbf{f}_t, \mathsf{Q}_t\right)$
  $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t := \text{UpdateEkf}\left(\overline{\boldsymbol{\mu}}_t, \overline{\boldsymbol{\Sigma}}_t, \mathbf{h}_t, \mathsf{R}_t, \mathbf{z}_t\right)$
  $t := t + 1$
end
return $\boldsymbol{\mu}_n$, $\boldsymbol{\Sigma}_n$

Algorithm 6: PredictEkf

Input: Belief distribution $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, Process model $\mathbf{f}$, Q
Output: Prior belief distribution $\overline{\boldsymbol{\mu}}$, $\overline{\boldsymbol{\Sigma}}$

$\overline{\boldsymbol{\mu}} := \mathbf{f}(\boldsymbol{\mu}, \mathbf{0})$
$\overline{\boldsymbol{\Sigma}} := \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \boldsymbol{\Sigma} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top + \frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}} \mathsf{Q} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}}^\top$
return $\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}$

The Extended Kalman Filter is given in Algorithm 5. Again, it comprises alternating prediction and update steps. The implementations of PredictEkf and UpdateEkf are given in Algorithms 6 and 7, respectively.

Note, that the EKF equations exactly resemble those of the linear Kalman Filter. The predicted state mean as well as the predicted measurement are obtained using the noise-free models. The Jacobian matrices $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ take the place the matrices A and H of the linear process and measurement models. Similarly, the linearised noise covariances $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}} \mathsf{Q} \frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}}^\top$ and $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}} \mathsf{R} \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}}^\top$ replace their counterparts Q and R. For a detailed derivation we refer to (Thrun et al., 2005, section 3.3.4).

It should be noted that in contrast to the Kalman Filter, the EKF only approximates the true belief distributions. Moreover, it is not possible to give any guarantees regarding the quality of this approximation, in general. The quality depends on the local nonlinearity of the models, i.e., how well models are locally approximated by linearisation.[3]

Nevertheless, "the EKF has become just about the most popular tool for state estimation in robotics. [...] EKFs have been applied with great success to a number of state estimation problems that violate the underlying assumptions." (Thrun et al., 2005, p. 61). Specifically, many implementations of visual SLAM rely on the EKF as the probabilistic inference mechanism.

The popularity of the EKF is in part due to its computational efficiency. In the general case, the complexity of one time step of the EKF is $\mathcal{O}(n^3 + m^3)$ where n is the dimension of

[3] Here, the notion of "local" depends on how sharply peeked the belief distribution is. In practice, it is therefore important to keep the belief uncertainty small.

Algorithm 7: UpdateEkf

Input: Prior belief distribution $\overline{\boldsymbol{\mu}}$, $\overline{\boldsymbol{\Sigma}}$, Measurement model $\mathbf{h}$, R, Measurement $\mathbf{z}$
Output: Posterior belief distribution $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

1 $\boldsymbol{\nu} := \mathbf{z} - \mathbf{h}\left(\overline{\boldsymbol{\mu}}, \mathbf{0}\right)$

2 $\mathsf{S} := \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \overline{\boldsymbol{\Sigma}} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top} + \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}} \mathsf{R} \frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}}^{\top}$

3 $\mathsf{K} := \overline{\boldsymbol{\Sigma}} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\top} \mathsf{S}^{-1}$

4 $\boldsymbol{\mu} := \overline{\boldsymbol{\mu}} + \mathsf{K}\boldsymbol{\nu}$

5 $\boldsymbol{\Sigma} := \overline{\boldsymbol{\Sigma}} - \mathsf{K}\mathsf{S}\mathsf{K}^{\top}$

6 **return** $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

the state vector and m is the dimension of the measurement vector. The cubic dependency on n stems from the multiplication of three $n \times n$ matrices in the term $\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \boldsymbol{\Sigma} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\top}$ in line 2 of PredictEkf (Algorithm 6). The cubic dependency on m stems from the inversion of the $m \times m$ matrix S in line 3 of UpdateEkf (Algorithm 7).

When applied to SLAM, the EKF complexity is reduced to $\mathcal{O}(n^2)$. This is due to the following structural assumptions made in the SLAM problem.

1. The environment is static. This allows to implement the prediction step in $\mathcal{O}(n)$ in the size of the state vector.

2. There is an upper bound on number of environment features simultaneously observed in every time step. This means that the measurement dimension m is bounded by a constant and consequently this makes the update step $\mathcal{O}(n^2)$ in the size of the state vector.

A detailed analysis of the EKF SLAM complexity can be found, e.g., in (Guivant & Nebot, 2001; Knight et al., 2001).

3.11. The Iterated Extended Kalman Filter

The linearisation of the models in the EKF is necessary because we want to represent the beliefs as Gaussians. Ideally, we would always linearise the models about the true state. However, the true state is not known. Instead, the EKF uses the mean of the belief distribution as the linearisation point.

In this section we describe the Iterated Extended Kalman Filter (Gelb, 1974, p. 190f), which addresses this problem. The IEKF is a simple modification of the EKF that replaces the standard update by an iterative procedure. As indicated above, the basic motivation of this iterated update step is that linearisation errors would be reduced if the linearisation point would be the true mean of posterior density. The posterior estimate of the EKF is closer to the true state than the prior estimate, and thus, linearisation should rather be

performed around the posterior estimate. Hence, the IEKF repeats the update step several times, where in each iteration the measurement model is re-linearised around the posterior estimate obtained in the previous iteration. The resulting procedure has been shown to be equivalent to using a Gauss-Newton method for optimizing the Kalman update (Bell & Cathey, 1993).

In the $(k+1)$th iteration the IEKF uses the posterior mean estimate of the previous iteration, $\boldsymbol{\mu}^k$, as the linearisation point for the measurement model. The initial linearisation point for the first iteration is the prior belief mean, $\boldsymbol{\mu}^0 = \overline{\boldsymbol{\mu}}$. If only this first iteration is performed then the IEKF is identical to the EKF.

Given $\boldsymbol{\mu}^k$ as a linearisation point, the first order Taylor expansion of the measurement model (3.56) yields

$$\begin{aligned}\mathbf{z}_t &= \mathbf{h}_t(\mathbf{x}_t, \boldsymbol{\delta}_t) \\ &\approx \mathbf{h}_t(\boldsymbol{\mu}^k, \mathbf{0}) + \left.\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_t}\right|_{(\boldsymbol{\mu}^k, \mathbf{0})} (\mathbf{x}_t - \boldsymbol{\mu}^k) + \left.\frac{\partial \mathbf{h}_t}{\partial \boldsymbol{\delta}_t}\right|_{(\boldsymbol{\mu}^k, \mathbf{0})} \boldsymbol{\delta}_t. \end{aligned} \tag{3.61}$$

This is identical to Equation 3.59 for the EKF linearisation, except that the expansion is performed at $\boldsymbol{\mu}^k$ rather than at the *a priori* state estimate $\overline{\boldsymbol{\mu}}_t$. The linearised model is then used in the Kalman filter update equations to obtain the posterior $\boldsymbol{\mu}^{k+1}$ which is used as the linearisation point for the next iteration. The resulting UpdateIekf step is presented in Algorithm 8.

The update iteration is performed in the loop in lines 3–11. The update process is repeated until the posterior estimate has converged or a predefined maximum number of iterations is exceeded. Before the first iteration the linearisation point is set to the prior estimate, i.e., $\boldsymbol{\mu}^0 = \overline{\boldsymbol{\mu}}$.

Let us compare the calculations within one iteration to the standard UpdateEkf calculation. Besides the recomputation of the Jacobian matrices, the only difference occurs in the computation of $\boldsymbol{\nu}$ in line 6. The innovation $\boldsymbol{\nu}$ is the difference between the actual measurement and the predicted measurement. Here, the predicted measurement is computed using the current linearisation of the measurement model. That is, the predicted measurement is $\mathbf{h}(\boldsymbol{\mu}^k, \mathbf{0}) + \mathsf{H}\left(\overline{\boldsymbol{\mu}} - \boldsymbol{\mu}^k\right)$. Note, that in the first iteration we have $\overline{\boldsymbol{\mu}} = \boldsymbol{\mu}^0$ and thus the predicted measurement is identical to the EKF.

The prediction step of the IEKF is the standard PredictEkf step.

Because the number of iterations in each UpdateIekf step is limited by the constant *max iterations*, the computational complexity of the IEKF remains the same as for the EKF. The computational overhead in practical implementations is often moderate. For example, in the SLAM setting, the most expensive step is the update of the covariance matrix in line 13 of the algorithm. This is computed only once, after the mean estimate has converged.

Algorithm 8: UpdateIekf

Input: Prior belief distribution $\overline{\boldsymbol{\mu}}$, $\overline{\boldsymbol{\Sigma}}$, Measurement model $\mathbf{h}$, R, Measurement $\mathbf{z}$
Output: Posterior belief distribution $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

$k := 0$
$\boldsymbol{\mu}^0 := \overline{\boldsymbol{\mu}}$
repeat
  $\mathsf{H} := \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{(\boldsymbol{\mu}^k, \mathbf{0})}$
  $\mathsf{G} := \left.\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}}\right|_{(\boldsymbol{\mu}^k, \mathbf{0})}$
  $\boldsymbol{\nu} := \mathbf{z} - \mathbf{h}(\boldsymbol{\mu}^k, \mathbf{0}) - \mathsf{H}\left(\overline{\boldsymbol{\mu}} - \boldsymbol{\mu}^k\right)$
  $\mathsf{S} := \mathsf{H}\overline{\boldsymbol{\Sigma}}\mathsf{H}^\top + \mathsf{G}\mathsf{R}\mathsf{G}^\top$
  $\mathsf{K} := \overline{\boldsymbol{\Sigma}}\mathsf{H}^\top\mathsf{S}^{-1}$
  $\boldsymbol{\mu}^{k+1} := \overline{\boldsymbol{\mu}} + \mathsf{K}\boldsymbol{\nu}$
  $k := k + 1$
until $|\boldsymbol{\mu}^k - \boldsymbol{\mu}^{k-1}| < \mathit{threshold}$ **or** $k \geq \mathit{max\ iterations}$
$\boldsymbol{\mu} := \boldsymbol{\mu}^k$
$\boldsymbol{\Sigma} := \overline{\boldsymbol{\Sigma}} - \mathsf{K}\mathsf{H}\overline{\boldsymbol{\Sigma}}$
return $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

4
The EKF Framework for Visual SLAM

4.1. Introduction

In this chapter we discuss a basic but complete visual SLAM algorithm that is built around the Extended Kalman Filter. With this, we follow two purposes. First, it serves as an overview of the tasks and challenges that must be addressed in a real-time implementation. Beyond the definition of state representation and probabilistic models for the EKF, the issues range from low-level image processing to high-level heuristic decision-making, e.g., about management of features in the map. Second, the algorithm presented in this chapter serves as a baseline implementation of visual SLAM. It provides the context in which the contributions of later chapters are developed and tested.

The implementation follows closely the system presented by Davison (2003). Because we use a stereo camera, appropriate extensions regarding stereo are made. However, from an algorithmic point of view the differences between the monocular and the stereo case are marginal. In fact, we try to keep the discussion generic with respect to the type of camera. We comment on the particularities of the stereo approach where appropriate.

Other differences to (Davison, 2003) include the application of the IEKF instead of the standard EKF, to reduce local linearisation errors. Moreover, the Joint Compatibility Branch and Bound (JCBB) algorithm by Neira & Tardos (2001) is incorporated. JCBB provides robust data association, allowing to detect erroneous feature measurements.

There are various steps in the algorithm which rely on heuristics. The details of these heuristics vary from implementation to implementation. In our description, we provide general thoughts and guidelines for these heuristics and describe our own implementation.

In the following Section 4.2 we state the underlying assumptions that are made in the presented visual SLAM framework. Section 4.3 presents a high-level overview and an outline of the EKF based visual SLAM algorithm. The remaining Sections 4.4 to 4.9 detail various aspects of the algorithm, the world representation, and the mathematical models employed in the EKF.

4.2. Assumptions

Visual SLAM approaches commonly rely on the following assumptions about the camera and the environment.

- The camera is calibrated, i.e., the intrinsic camera parameters are known.
- The camera is moving smoothly, i.e., there are no discontinuities in the trajectory.
- The scene is sufficiently textured. This is a natural restriction when using a camera as the only sensor.
- The environment is static, i.e., nothing is moving except the camera.

Clearly, these assumptions are sometimes violated in real world scenarios. Various methods have been proposed in the literature to increase robustness with respect to such violations. For example, re-localisation approaches allow recovery from tracking failure that may be caused by jerky camera motion or featureless areas in the scene. The basic algorithm presented here we will not consider such enhancements. However, these extensions would be required to make the algorithm function robustly under adverse conditions.

4.3. Overview

The task in visual SLAM is to infer from a sequence of camera images the pose of the camera and a map of the environment. The focus is on real-time, sequential operation: At every instant, we are interested in the best guess of the camera pose and map given the currently available data. In the EKF framework, visual SLAM is cast as a state estimation task. The camera pose and the environment map are encoded in a joint state vector. We define a process model, which describes the camera motion, as well as a generative measurement model, which explains observations of map entities in the camera image as functions of the state vector. At every instant, the current belief about the state of the world is encoded as a Gaussian distribution on the state space. The EKF provides the probabilistic machinery to propagate the belief distribution through time and perform updates with new information from image measurements.

Around this probabilistic core a system is build that controls the map building process and provides an interface to the raw image data. This includes additional data, e.g., concerning the visual appearance of map features, and additional procedures, e.g., for deleting old or adding new features to the state vector.

In the following, we give a high-level overview of this EKF-based visual SLAM system, which is then followed by a more formal definition of the algorithm. As indicated above, the current world state is represented as joint state vector. The state vector $\mathbf{x}$ can be partitioned into a part describing the camera and a part describing the map.

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_v \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{pmatrix} \in \mathbb{R}^N \tag{4.1}$$

The block $\mathbf{x}_v$ describes the current camera pose and velocity. The remaining elements of the state vector describe the map. The map comprises a sparse set of landmarks or *features* $\mathbf{y}_1 \dots \mathbf{y}_n$. Specifically, here we consider *point features*, which are employed by the majority of visual SLAM systems. A point feature is a specific fixed 3D point in the environment. The point (or rather a small local region around it) should be visually salient, such that its appearance can be used to locate the point in images. In the state vector, $\mathbf{y}_i$ simply comprises the 3D world coordinates of the ith feature point.[1]

The belief about the joint state $\mathbf{x}$ is modeled as a multivariate Gaussian distribution. The distribution is represented by its mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

$$\text{bel}(\mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4.2}$$

As introduced in Section 3.8, the belief is the probability distribution of the state $\mathbf{x}$ at time t conditioned on the measurements made up to t. Thus, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are time dependent. However, to unclutter notation we omit time indices in the following.

Equivalently to the state, the mean and covariance can be partitioned into camera and feature blocks.

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_{\mathbf{x}_v} \\ \boldsymbol{\mu}_{\mathbf{y}_1} \\ \vdots \\ \boldsymbol{\mu}_{\mathbf{y}_n} \end{pmatrix}, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{x}_v\mathbf{x}_v} & \boldsymbol{\Sigma}_{\mathbf{x}_v\mathbf{y}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{x}_v\mathbf{y}_n} \\ \boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{x}_v} & \boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{y}_1\mathbf{y}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Sigma}_{\mathbf{y}_n\mathbf{x}_v} & \boldsymbol{\Sigma}_{\mathbf{y}_n\mathbf{y}_1} & \cdots & \boldsymbol{\Sigma}_{\mathbf{y}_n\mathbf{y}_n} \end{bmatrix}. \tag{4.3}$$

The diagonal covariance matrix blocks $\boldsymbol{\Sigma}_{\mathbf{x}_v\mathbf{x}_v}$ and $\boldsymbol{\Sigma}_{\mathbf{y}_i\mathbf{y}_i}$ express the marginal uncertainty about the camera or a feature, respectively. Off-diagonal blocks $\boldsymbol{\Sigma}_{\mathbf{y}_i\mathbf{y}_j}$ and $\boldsymbol{\Sigma}_{\mathbf{x}_v\mathbf{y}_i}$ describe correlation between two features or between a feature and the camera, respectively.

The joint belief is updated sequentially using the predict-update cycle of the EKF. For this, we have to describe the system by defining a process model and a measurement model. New images are captured by the camera at a frame rate of 30 Hz. The process model is used to predict how the state evolves during the 33 ms period of "temporary blindness" between images. The measurement model describes how the observations, i.e., the camera images, are generated from the state.

For the process model we consider the challenging case where the camera is held in hand and moved around by a human operator. In this scenario, the camera motion is beyond our control and it is impossible to accurately predict how the camera pose will evolve. However, it is also clear that the camera cannot change position completely at random from one frame to the next. The camera will not jump from one corner of the room to another. Instead, we can expect a certain smoothness in the camera trajectory. We use a *constant-velocity* process model. This model describes the camera as moving with constant velocity and constant angular velocity – in the absence of external forces causing acceleration or angular acceleration. It is further assumed that such undetermined accelerations occur with a known Gaussian profile. Accelerations are modeled as process noise.

[1] This is the straightforward Euclidean representation of map features. Because it results in simple parameterisation as well as simple measurement and initialisation models, we will employ it for the purpose of this overview despite its well-known problems.

The measurement model describes how the measurement is generated from the state. Our sparse point map representation is too abstract to really use the raw image data as measurements. Instead, a measurement of a feature $\mathbf{y}_i$ consists of the image coordinates of its projection in the current frame.

These feature measurements are obtained from the images by template matching. With each feature, a small template patch (e.g., 11×11 pixels) of its appearance is stored. In subsequent images, measurements of this feature are made by searching for the pixel window with maximal correlation to this template. While a certain degree of bottom-up computation is necessary to obtain measurements from the images, this can be guided by top-down information in a process referred to as *active search*.

So far we have implicitly assumed that the state vector comprises n features at any time. However, frequently we need to add new features to the state because the camera is exploring previously unseen parts of the environment. In fact, in the first frame we start out with zero features in the state vector. We also want to delete features sometimes, for instance, because it turns out that they do not correspond to static parts of the scene. The state vector grows or shrinks accordingly.

Finally, in a real-time setting decisions have to be made as to how to best utilize the tightly restricted time budget available per camera frame. These decisions occur at various points in the algorithm and are usually based on heuristics. For example, if the state contains too many features the $\mathcal{O}(n^2)$ EKF update will break the real-time constraint. If the camera is constantly exploring new territory this is inevitable at some point. However, a clever strategy to decide when and where to (sparsely) initialise new features may postpone this point and increase the size of environments that can be handled. Also, in general there is not enough time to make measurements of *all* features that are potentially visible in a given frame. Thus we have to decide which feature measurements should be attempted.

At this point we want to give a more formal description of the algorithm. The top-level VisualSlam procedure is given in Algorithm 9. It provides an overview of the steps that are performed for every camera image. The individual steps will be elaborated in the subsequent sections.

The algorithm starts with the initialisation of the internal data structures that represent the world state and are subsequently updated with every new camera image. We distinguish between the *probabilistic state* and the *auxiliary state*.

The probabilistic state consists of the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ which represent the joint belief (4.2) about the state of the camera and the map, i.e., about the state vector $\mathbf{x}$. The specific parameterisation of the camera and the map features will be given in Section 4.4. In line 1, the probabilistic state is initialised by the InitState function. The initial $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ describe a camera pose in the origin of the world coordinate system with no uncertainty. The initial map is empty.

The auxiliary state $\mathcal{P}$ comprises data that is not part of the probabilistic state but nevertheless required for low-level image processing and various heuristics. In contrast to the probabilistic state it is updated separately from the EKF. In particular, $\mathcal{P}$ comprises feature appearance A and measurement heuristics S. Besides that, various constants such as the internal calibration parameters of the camera are part of the auxiliary state.

$A = (a_1, \ldots, a_n)$ is the *feature appearance* data. It is an array that stores for every

Algorithm 9: VisualSlam

Output: Final state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$

```
1  μ, Σ := InitState
2  A, S := ∅, ∅
3  I, Δt := GrabImage
4  μ, Σ, A, S := ReviseMap (μ, Σ, I, A, S)
5  while running do
6  |  I, Δt := GrabImage
7  |  μ, Σ := PredictState (μ, Σ, Δt)
8  |  M, S := MeasureFeatures (μ, Σ, I, A, S)
9  |  μ, Σ := UpdateState (μ, Σ, M)
10 |  μ, Σ, A, S := ReviseMap (μ, Σ, I, A, S)
11 end
12 return μ, Σ
```

feature pose $\mathbf{y}_i$ the appearance a_i of the corresponding feature. The appearance a_i consists of any information required to make image measurements of the ith feature. In our case, the appearance of a feature contains a 21×21 pixel template of the neighbourhood of the feature in the image where the feature was initially observed. Other implementations might encode the appearance as a SIFT descriptor (Lowe, 2004). Additional information might be used, such as visibility information (Wuest et al., 2007). For the purpose of this overview we will leave these details unspecified.

$S = (s_1, \ldots, s_n)$ is the *measurement statistics* data. It is an array that stores, for every feature $\mathbf{y}_i$ in the probabilistic state, statistic information s_i collected during measurement attempts so far. For, instance the statistics contains the ratio of attempted to failed measurements which can be used to detect spurious features that should be deleted from the map.

A and S are initialised in line 2 of the algorithm. They are both initially empty because the map contains no features yet. In lines 3 and 4 an image is obtained from the camera and the first features are initialised into the map. The functions GrabImage and ReviseMap which are employed will be discussed shortly

The main part of the algorithm is the **while** loop in lines 5 – 11. The loop is iterated while new images should be processed. After the loop is left the algorithm returns the final state estimate and terminates.[2]

Let us look at the statements in the body of the loop.

- GrabImage in line 6 captures a new image from the camera. It returns the image data I and the time Δt in seconds that has passed since the last image was captured,

[2] In practice, we are often not only interested in this final estimate. Instead, the VisualSlam algorithm will be integrated into a larger system and provide up-to-date state estimates for every image, i.e., the VisualSlam main loop provides $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ after each iteration

i.e., since the last invocation of GrabImage. Algorithm 9 generically applies to both monocular and stereo settings. In the stereo case, the image data I is taken to comprise the stereo pair $I = (I_r, I_l)$.

- PredictState in line 7 encapsulates the EKF predict step. We predict how the state evolved during the time that passed since the last image. Usually this is done using a constant velocity or constant position motion model. PredictState uses the Δt parameter to instantiate the motion model and then applies the EKF prediction equations. The updated estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ now encodes the current belief without having looked at the current image I, yet.

- MeasureFeatures in line 8 takes measurements of some map features in the current image. The current state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ is used to determine which features we will attempt to measure in the current image. These features are then searched for in the image I. This is accomplished using the feature appearance data A. If the image search for a feature is successful, the image position is a 2D (3D in the stereo case) measurement vector for the feature. The feature measurement statistics are updated according to successful or failed measurement. MeasureFeatures returns a list of successful measurements M and the updated measurement statistics S.

- UpdateState in line 9 updates the state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ with the new measurement information. A measurement model is instantiated according to the list of measurements M. Then we apply the EKF (respectively IEKF) update equations.

- Finally, ReviseMap in line 10 performs map maintenance tasks. Spurious features are removed from the map. The heuristic that decides which features to remove is based on the measurement statistics S. New features are initialised into the probabilistic state as required. Their appearance is stored in A. ReviseMap returns the revised state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, appearance data A, and statistics data S.

We will provide details on these steps in the following sections. In the next section we describe the parameterisation of the camera and the map in the state vector. In Section 4.5, we detail the PredictState function and discuss the constant-velocity motion model that is used in our implementation. In Section 4.6, we elaborate the UpdateState function and discuss the feature measurement model. Then, in Section 4.7, we detail the MeasureFeatures function and describe how measurements are actually made in the camera images. We describe the template matching procedure used to measure features in the current image. We discuss the process of active search and the heuristic that decides which features to measure. Finally, in Section 4.8, we elaborate the ReviseMap function. We discuss map management heuristics and the probabilistic state calculations required to intialise or delete features.

4.4. State Vector Parameterisation

The probabilistic state vector $\mathbf{x}$ comprises those aspects of the camera and the map that are modeled in the belief distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and are updated by the EKF. This includes

the 3D pose of the camera and the 3D coordinates of feature points. Other aspects, such as the internal calibration parameters of the camera and appearance templates of feature points are handled independently from the EKF. They are stored in the auxiliary state $\mathcal{P}$ and are either assumed fixed or updated using independent mechanisms. We will not address those aspects here. Instead, we focus on the representation of the camera and the map in the state vector $\mathbf{x}$.

4.4.1. Camera Pose and Velocity

The current camera state is described by the $\mathbf{x}_v$ block of the probabilistic state. Similar to Davison (2003) we model the camera state as

$$\mathbf{x}_v = \begin{pmatrix} \mathbf{r}^{\mathcal{WC}} \\ \mathbf{q}^{\mathcal{WC}} \\ \mathbf{v}^{\mathcal{W}} \\ \boldsymbol{\omega}^{\mathcal{W}} \end{pmatrix} \in \mathbb{R}^{13}. \tag{4.4}$$

The first two blocks describe the current camera pose, i.e., the pose $p^{\mathcal{WC}}$ of the camera coordinate frame $\mathcal{C}$ measured in the world coordinate frame $\mathcal{W}$. The translation 3-vector $\mathbf{r}^{\mathcal{WC}}$ is the position of the camera center within the world coordinate frame, and the quaternion rotation $\mathbf{q}^{\mathcal{WC}}$ is the relative rotation between the camera and world coordinate frames. In the pose notation introduced in Section 3.2, we write

$$p^{\mathcal{WC}} = (\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}, \mathbf{r}^{\mathcal{WC}}) \tag{4.5}$$

where $\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}$ denotes the rotation matrix corresponding to $\mathbf{q}^{\mathcal{WC}}$.

The remaining two blocks of $\mathbf{x}_v$ describe the translational velocity and the angular velocity of the camera. The 3-vector $\mathbf{v}^{\mathcal{W}}$ represents the rate of linear motion along the three axes of the world coordinate frame in meters per second. The 3-vector $\boldsymbol{\omega}^{\mathcal{W}}$ represents the angular velocity in angle-axis form: The direction of $\boldsymbol{\omega}^{\mathcal{W}}$ is the axis of rotation in the world frame, and its vector norm is the rate of rotation in radians per second.

The reason for making the velocities part of the probabilistic state lies in the Markov assumption made in the process model of the Bayes Filter. The Markov assumption implies that the present state must contain everything that is needed to predict the next state – knowledge of past states and measurements provides no additional information. We want to use a constant-velocity model of camera motion. Thus, the camera's current velocity must be represented in the state.

4.4.2. Map Features

The second part of the state vector comprises the map. The map is a set of features $\mathbf{y}_1, \ldots, \mathbf{y}_n$. For now, we can simply consider a feature as a fixed 3D point in the environment. To be able to make measurements of such a point feature, additional information is needed about how the feature appears in the camera image. However, this information is not modeled as a part of the probabilistic state.

Algorithm 10: PredictState

Input: Prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, Set of measurements M
Output: Posterior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$

1 $\mathbf{f} :=$ BuildMotionModel (Δt)
2 $\boldsymbol{\mu}, \boldsymbol{\Sigma} :=$ PredictEkf $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{f}, \mathsf{Q})$
3 **return** $\boldsymbol{\mu}, \boldsymbol{\Sigma}$

A feature is parameterised in the state vector by its coordinates in the world frame $\mathcal{W}$,

$$\mathbf{y}_i = \begin{pmatrix} x_i^{\mathcal{W}} \\ y_i^{\mathcal{W}} \\ z_i^{\mathcal{W}} \end{pmatrix}. \tag{4.6}$$

This representation of point features is referred to as the *Euclidean* parameterisation. It was initially used in the system by Davison (2003) and similar systems. It is appealing because it provides straightforward measurement and initialisation equations and because with 3 parameters it is a relatively efficient parameterisation in terms of state size. However, it is a well-known problem that only features relatively close to the camera can be handled. The uncertainty arising for distant features is not well-represented as a Gaussian distribution in Euclidean XYZ space. This is because the relationship between the depth of a feature and the associated parallax in the image is highly non-linear. Inverse depth representations have been proposed to address this problem (e.g., Montiel et al., 2006). We will come back to this issue later. For the purposes of this introduction however, we will stick with the simple Euclidean model.

4.5. Process Model and Prediction Step

The prediction step of the EKF uses a process model to propagate the probabilistic state forward in time. The process model predicts how the state evolves in the period between the previous and the current image.

In the VisualSlam algorithm, this is realised by the PredictState step in line 7. PredictState is detailed in Algorithm 10. The algorithm consists of two steps. First the process model is instantiated with the time Δt that has passed since the previous image. Then this process model is used in the EKF prediction step.

In the following, we formulate a nonlinear process model as discussed in Section 3.10, by describing frame-to-frame camera motion as a state transition function of the form (3.55). The process model is parameterised on Δt. Here, Δt denotes the time in seconds passing between time-steps $t-1$ and t, i.e., between capturing the previous and the current camera image. In the absence of accelerations, the camera moves with constant velocity due to inertia. The current camera pose at time-step t is determined by the previous camera pose and velocity at time-step $t-1$. Assuming that the camera moves with constant linear and

angular velocity, we have

$$\mathbf{r}_t^{\mathcal{WC}} = \mathbf{r}_{t-1}^{\mathcal{WC}} + \Delta t \cdot \mathbf{v}_{t-1}^{\mathcal{W}} \tag{4.7}$$

$$\mathbf{q}_t^{\mathcal{WC}} = \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}}\right) \circ \mathbf{q}_{t-1}^{\mathcal{WC}}. \tag{4.8}$$

The velocity remains constant over time

$$\mathbf{v}_t^{\mathcal{W}} = \mathbf{v}_{t-1}^{\mathcal{W}} \tag{4.9}$$

$$\boldsymbol{\omega}_t^{\mathcal{W}} = \boldsymbol{\omega}_{t-1}^{\mathcal{W}}. \tag{4.10}$$

Suppose that a linear acceleration $\mathbf{a}_t$ and angular acceleration $\boldsymbol{\alpha}_t$ occur during the time-step. These unknown accelerations can be collected in the noise vector

$$\boldsymbol{\varepsilon}_t = \begin{pmatrix} \mathbf{a}_t \\ \boldsymbol{\alpha}_t \end{pmatrix} \in \mathbb{R}^6. \tag{4.11}$$

We assume that this process noise is temporally white and zero-mean Gaussian distributed with covariance Q_t. The noise covariance matrix Q_t characterises the accelerations we expect on average. This covariance is assumed to be known and must be tuned to the particular application, e.g., we would expect very different acceleration profiles for a hand-held camera being moved by a careful human operator and a camera attached to a football that is kicked around. In our implementation, we use a predefined constant Q matrix that is part of the auxiliary program state $\mathcal{P}$.

To obtain the process model, we incorporate the accelerations in the prediction equations (4.7)-(4.10). Let us further remember that the environment, and therefore the features, are assumed to be static. We can write the state transition function $\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t;\, \Delta t)$ as

$$\begin{pmatrix} \mathbf{r}_t^{\mathcal{WC}} \\ \mathbf{q}_t^{\mathcal{WC}} \\ \mathbf{v}_t^{\mathcal{W}} \\ \boldsymbol{\omega}_t^{\mathcal{W}} \\ \vdots \\ \mathbf{y}_{i;t} \\ \vdots \end{pmatrix} = \mathbf{f}(\mathbf{x}_{t-1}, \boldsymbol{\varepsilon}_t;\, \Delta t) = \begin{pmatrix} \mathbf{r}_{t-1}^{\mathcal{WC}} + \Delta t \cdot \mathbf{v}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2} \cdot \mathbf{a}_t \\ \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2} \cdot \boldsymbol{\alpha}_t\right) \circ \mathbf{q}_{t-1}^{\mathcal{WC}} \\ \mathbf{v}_{t-1}^{\mathcal{W}} + \Delta t \cdot \mathbf{a}_t \\ \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \Delta t \cdot \boldsymbol{\alpha}_t \\ \vdots \\ \mathbf{y}_{i;t-1} \\ \vdots \end{pmatrix}. \tag{4.12}$$

This process is instantiated with the concrete Δt in line 1 of Algorithm 10. In the EKF predict step the model is linearised, cf. Equation 3.57. Therefore, we need the Jacobian matrices, i.e., the partial derivatives $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}}$ evaluated at $\mathbf{x}_{t-1} = \boldsymbol{\mu}_{t-1}$, $\boldsymbol{\varepsilon}_t = \mathbf{0}$. These Jacobians can be found in Appendix A.3.

4.6. Measurement Model and Update Step

The update step of the EKF uses a measurement model to incorporate new observations into the state estimate. The measurement model describes how the measurements are

Algorithm 11: UpdateState

Input: Prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, Set of measurements M
Output: Posterior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$

1 $\mathbf{h}, \mathsf{R}, \mathbf{z} :=$ BuildMeasurementModel (M)
2 $\boldsymbol{\mu}, \boldsymbol{\Sigma} :=$ UpdateIekf $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{h}, \mathsf{R}, \mathbf{z})$
3 $\boldsymbol{\mu}, \boldsymbol{\Sigma} :=$ NormaliseQuaternion $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
4 **return** $\boldsymbol{\mu}, \boldsymbol{\Sigma}$

"generated" from the current state. In the VisualSlam algorithm, this is realised by the UpdateState step in line 9. UpdateState is presented in Algorithm 11.

In every camera image, we measure the projections of some of the currently visible features. The set of measurements is the input M of the UpdateState algorithm. M is a list of pairs $(i, \mathbf{z}_i)$ where i is the index of a feature in the state vector and $\mathbf{z}_i$ is the measurement of that feature. In the monocular case, $\mathbf{z}_i \in \mathbb{R}^2$ are the observed image coordinates of the projection of feature $\mathbf{y}_i$. In the stereo case, $\mathbf{z}_i \in \mathbb{R}^3$ is the stereo projection.

The BuildMeasurementModel step stacks the measurements into the joint measurement vector $\mathbf{z}$ and constructs the corresponding generative measurement model $\mathbf{h}$ and measurement noise covariance R.

Then we use this information to update the probabilistic state in line 2 of the algorithm. In our implementation we use the IEKF update as introduced in Section 3.11. It has been shown by Tully et al. (2008) that applying an iterated filter to the problem of bearing-only SLAM increases map accuracy in comparison to the standard EKF update.[3]

Finally, the NormaliseQuaternion step is performed. This step is required because the linearised measurement model cannot express the unit length constraint on the quaternion $\mathbf{q}^{\mathcal{WC}}$ representing camera rotation. This constraint is explicitly enforced after each update step. The quaternion in the mean vector is normalised to unit length and the covariance matrix is updated accordingly.

In the following, we formulate a generative measurement model as discussed in Section 3.10, by describing the measurements as a function of the form (3.56). Without loss of generality, let us assume that the features $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are measured in the current image. Let $\mathbf{y}_i^{\mathcal{I}}$ denote the (perfect) projection of feature $\mathbf{y}_i$ in the image. The image measurements $\mathbf{z}_1, \ldots, \mathbf{z}_i$ of these projections are obtained by a template matching procedure which will be described in detail in the next section. There are several sources of measurement noise, such as intensity noise in the image, discretization of the pixel coordinates and intensities, unmodeled local scene geometry, etc. We assume that, per feature, these effects of noise can be subsumed in an additive Gaussian error term $\boldsymbol{\delta}_i$, and that these errors occur uncorrelated between different feature measurements.

[3] Tully et al. (2008) also incorporate backtracking search in each EKF iteration to take a partial step in the Newton direction and can thus guarantee that the new estimate in every iteration reduces the cost. Here, we follow the simpler approach of the original IEKF (Gelb, 1974) which is equivalent to taking a full Newton step in every iteration. This is considered risky by Tully et al. (2008) because taking a full step may increase the cost. However, in our experiments the approach has proven sufficient.

Then, we can write a generative measurement model of the form (3.56),

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \begin{pmatrix} \mathbf{y}_1^{\mathcal{I}} + \boldsymbol{\delta}_1 \\ \vdots \\ \mathbf{y}_k^{\mathcal{I}} + \boldsymbol{\delta}_k \end{pmatrix}. \tag{4.13}$$

For notational convenience we have omitted the time indices. The joint error $\boldsymbol{\delta}$ comprises the individual measurement errors $\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_k$. Because the individual errors are Gaussian and uncorrelated we have $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{R})$ where the covariance is a block-diagonal matrix of the individual covariances

$$\mathsf{R} = \begin{bmatrix} \mathsf{R}_1 & \ldots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathsf{R}_k \end{bmatrix}. \tag{4.14}$$

The per feature measurement covariance can be supplied with each measurement by the matching procedure. In our implementation, we simply use a predefined constant R_i matrix that is part of the auxiliary program state $\mathcal{P}$.

To find the coordinates $\mathbf{y}_i^{\mathcal{I}}$ of the projection of a feature in the current image, we first transform its world coordinates $\mathbf{y}_i$ into the current camera coordinate frame

$$\mathbf{y}_i^{\mathcal{C}} = p^{\mathcal{CW}}(\mathbf{y}_i \,; \mathbf{x}_v). \tag{4.15}$$

The resulting camera coordinates are then projected into the image

$$\mathbf{y}_i^{\mathcal{I}} = \pi_s\left(\mathbf{y}_i^{\mathcal{C}}\right) = \pi_s\left(p^{\mathcal{CW}}(\mathbf{y}_i \,; \mathbf{x}_v)\right). \tag{4.16}$$

In the above equations, we have used the notation $p^{\mathcal{CW}}(\cdot \,; \mathbf{x}_v)$ to emphasize that the pose transformation $p^{\mathcal{CW}}$ is parameterised by the current camera state.[4] We have used the stereo projection function $\pi_s(\cdot)$. In the monocular case this is simply replaced by $\pi(\cdot)$. Note that the projection functions require knowledge of the internal calibration parameters of the camera, which must be also supplied in the auxiliary program state $\mathcal{P}$.

Thus, we have the generative model for an individual feature measurement

$$\mathbf{z}_i = \mathbf{y}_i^{\mathcal{I}} + \boldsymbol{\delta}_i = \pi_s\left(p^{\mathcal{CW}}(\mathbf{y}_i \,; \mathbf{x}_v)\right) + \boldsymbol{\delta}_i. \tag{4.17}$$

Assembling the appropriate individual measurement functions into a joint model of the form (4.13) is carried out by BuildMeasurementModel according to the measurements that occur in M.

When we apply the measurement model (4.13) in the EKF update step we must linearise it (see Equation 3.59). We need the Jacobian matrices $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}}$ evaluated at $\mathbf{x} = \overline{\boldsymbol{\mu}}_t$, $\boldsymbol{\delta} = \mathbf{0}$. These Jacobians are easily worked out, as we show in Appendix A.1.

[4] With the camera parameters $\mathbf{r}^{\mathcal{WC}}$, $\mathbf{q}^{\mathcal{WC}}$ from the state vector we have

$$p^{\mathcal{CW}} = p^{\mathcal{WC}\,-1} = (\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1},\ -\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\,\mathbf{r}^{\mathcal{WC}}).$$

Algorithm 12: MeasureFeatures

Input: Prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, Current image I, Auxiliary state A, S
Output: Set of measurements M, Updated measurement statistics S

```
1  M := ∅
2  i := SelectCandidate(μ, Σ, A, M)
3  while i ≠ None do
4  |   z := MeasureFeature(μ, Σ, i, a_i, I)
5  |   M := M ∪ {(i, z)}
6  |   s_i := UpdateStatistics(s_i, z)
7  |   i := SelectCandidate(μ, Σ, A, M)
8  end
9  M, S := Jcbb(μ, Σ, M, S)
10 return M, S
```

4.7. Feature Measurement Using Active Search

The MeasureFeatures step in line 8 of the VisualSlam algorithm provides a set of feature measurements in the current image. MeasureFeatures is described in Algorithm 12.

The input to the algorithm is the prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, the current image I, the feature appearance data A, and the measurement statistics data S. The output is a set of successful measurements M and the updated measurement statistics. M is a set of pairs $(i, \mathbf{z})$ where i is the index of a feature in the state vector and $\mathbf{z}$ is a measurement.

In the main loop, starting from line 3, candidate measurements are attempted one by one. In every iteration, SelectCandidate returns the index i of the next candidate. This step employs a heuristic to select a feature to measure by considering one or more of the following: the expected visibility of the feature, the expected information content of the measurement, the distribution of already completed measurements in the image, the processing time already spent on the current image. Details about the SelectCandidate implementation in our system are provided in Section 4.7.3.

The measurement of the image coordinates of the feature is accomplished by the MeasureFeature function which will be described presently. It returns the measurement $\mathbf{z} \in \mathbb{R}^2 \cup \{\textit{Failure}\}$ in the monocular case, respectively $\mathbf{z} \in \mathbb{R}^3 \cup \{\textit{Failure}\}$ in the stereo case.

The index-measurement pair $(i, \mathbf{z})$ is added to the set of measurements in line 5 of the algorithm. The measurement statistics for the ith feature are updated according to the outcome of the measurement attempt by the UpdateStatistics function. Then the next measurement candidate is selected.

Finally, after all candidates have been measured the Jcbb function is invoked in line 9. Jcbb evaluates the *joint compatibility* of the measurements in M. The notion of joint compatibility was introduced by Neira & Tardos (2001) as a criterion for assessing the

Algorithm 13: MeasureFeature

Input: Prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, Feature index i, Feature appearance a_i, Current image I

Output: Measurement $\mathbf{z}$

1 $J :=$ PredictTemplate $(\boldsymbol{\mu}, i, a_i)$

2 $R :=$ PredictSearchRegion $(\boldsymbol{\mu}, \boldsymbol{\Sigma}, i)$

3 $\mathbf{z} :=$ TemplateMatch (J, I, R)

4 **return z**

quality of data association hypotheses. Correct data association, i.e., correctly associating sensor observations and map features, is one of essential parts of EKF based SLAM. The active search paradigm used in visual SLAM provides correct feature associations most of the time. Image features have distinctive appearance, and prior knowledge is used to tightly confine the image region to be considered for matches. Incorrect associations are rare. They occur chiefly due to moving objects in the scene or repeated texture. Nevertheless, even very few incorrect associations can corrupt the quality of an otherwise good map, as recently demonstrated by Clemente et al. (2007).

The joint compatibility test provides a method to detect erroneous feature matchings. The underlying idea is to consider constraints *between* measurements that also arise from prior information. The whole set of measurements accepted in the current image must be jointly consistent with the prior information.

Jcbb selects and returns the largest jointly compatible subset of successful measurements from M. In particular, we employ the Joint Compatibility Branch and Bound algorithm (Neira & Tardos, 2001) in the simplified form described by Clemente et al. (2007). The details are beyond the scope of this thesis. Jcbb updates the measurement statistics S according to the acceptance or rejection of individual measurements. Note, that after Jcbb the final M contains only successful measurements.

Now, lets look at the core step of the algorithm: the MeasureFeature function. It is described in Algorithm 13. MeasureFeature attempts a measurement of the ith feature in the image I. With every feature a small patch of pixels is stored as a template for its appearance. This template is taken from the first observation of the feature, and is used to make measurements of the feature in subsequent images. It is supplied as (part of) the feature appearance a_i. Together with the prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ this can be used to make predictions about the appearance in the current image and the image region where the feature will be found.

The image measurement process is based on template matching, which is performed by the TemplateMatch step in the algorithm. Measurements are made by template matching, that is, by searching the image for a *matching template* J of the feature. The matching template is a small image patch of how we expect the local neighbourhood of the feature point to appear in the current camera image. The image is searched for the best match to this template (the most similar pixel patch within the search region). The position of the best match provides the 2D image position of the feature.

Template matching requires that we are able to state how we expect the feature to appear in a given image. This is handled in line 1 of Algorithm 13. PredictTemplate creates the matching template J from the appearance information a_i and the best guess of the state vector $\boldsymbol{\mu}$. How this is accomplished depends on the specific assumptions about the environment and the information available. The simplest assumption would be that the appearance of the feature does not change at all: The reference template patch for the feature is taken from the camera image when the feature is initialised, and this patch is used to match the feature in all subsequent images. This approach was for instance employed in Davison's original system (Davison, 2003). In this case, PredictTemplate simply returns the reference template from a_i. Clearly, more sophisticated approaches are conceivable but for now we will not go into details. We will return to the issue of appearance prediction in Chapter 6.

A common characteristic of many visual SLAM systems is the active, top-down approach to search. Exhaustive template matching is computationally expensive: The best match to is found by scanning the template across the image and computing, at every pixel, some similarity measure. Taking prior belief into account can drastically reduce the image area that has to be searched. Thus, efficiency can be increased by focussing processing resources in image areas where matches will be found with high probability. At the same time robustness is improved because smaller search regions reduce the chance of mismatching with unrelated but visually similar image features. In line 2 of the algorithm the PredictSearchRegion function is invoked. PredictSearchRegion computes the uncertainty about the feature position in the image from the prior state estimate. Gating at 0.99 probability yields an elliptic search region R in image space. In the stereo case, the search region is an ellipsoid in (u, v, d) space.[5] Details on active search will be discussed in Section 4.7.2.

The final step of the MeasureFeature algorithm is the actual image search: TemplateMatch. The best match to the template J is searched for in region R in the current image I. If the difference between the best match and the template is smaller than a threshold, the measurement is deemed successful and the position of the best match is returned. Otherwise, the measurement fails and *Failure* is returned. The stereo case is handled analogously, except that the template is searched for in both images. Details on the similarity measure and the matching procedure are given in the next section.

4.7.1. Template Matching

We will first describe the dissimilarity measure used in our implementation. Then we discuss the exhaustive search which gives a pixel-accurate matching position. Optionally, this may be refined to subpixel accuracy as described subsequently. These methods are presented in a monocular setting first. Finally, we address how they are employed in the stereo case.

[5] This ellipsoid can be projected then into the left and right image to obtain individual search regions.

Zero-mean Sum of Squared Differences

Consider the matching template J. Let $X \subset \mathbb{R}^2$ be the set of pixels coordinates in the template. Then we can describe the template as an image function $J : X \to \mathbb{R}$. Similarly, let $I : \mathbb{R}^2 \to \mathbb{R}$ be the input image.

Note, that for simplicity we assume that the input image is infinitely large. Moreover, we assume that the image function is well-defined at non-integer pixel locations. The real camera image is discretized. Image values at non-integer pixel positions are interpolated using a suitable method. Specifically, our implementation uses bilinear interpolation.

The sum of squared differences between the image and the template placed at position $\mathbf{p}$ is defined as

$$\mathrm{ssd}\left(I, J, \mathbf{p}\right) = \sum_{\mathbf{x} \in X} \left(J\left(\mathbf{x}\right) - I\left(\mathbf{x} + \mathbf{p}\right)\right)^2 . \tag{4.18}$$

This provides a measure of dissimilarity between the images.

To make the measure robust to intensity variations we normalise the images such that the mean intensities are zero, where the mean is computed over the region covered by the template. This gives the *zero-mean sum of squared differences* (ZSSD) measure which is defined as

$$\mathrm{zssd}\left(I, J, \mathbf{p}\right) = \sum_{\mathbf{x} \in X} \left(\left(J\left(\mathbf{x}\right) - \bar{J}\right) - \left(I\left(\mathbf{x} + \mathbf{p}\right) - \bar{I}\right)\right)^2 \tag{4.19}$$

with the mean intensities

$$\bar{J} = \frac{1}{N} \cdot \sum_{\mathbf{x} \in X} J\left(\mathbf{x}\right) \tag{4.20}$$

$$\bar{I} = \frac{1}{N} \cdot \sum_{\mathbf{x} \in X} I\left(\mathbf{x} + \mathbf{p}\right) \tag{4.21}$$

where N is the number of pixels in X.

SSD based cost functions are widely used in feature tracking and stereo matching (Shi & Tomasi, 1994; Baker & Matthews, 2004; Nickels & Hutchinson, 2002; Hermann & Klette, 2009). SSD, normalised SSD, and ZSSD are also the standard dissimilarity measures used in feature-based visual SLAM. The ZSSD has been used for instance by Klein & Murray (2007).

Finding the best match

Let $R \subset \mathbb{R}^2$ be the set of pixel coordinates in the search region. The best match to the template image is found using exhaustive search. The template J is placed at every pixel in the search region and the ZSSD is computed. The pixel where the ZSSD is minimal is the best match. The objective function minimised by exhaustive search is

$$\mathbf{p}^* = \underset{\mathbf{p} \in R}{\arg\min}\ \mathrm{zssd}\left(I, J, \mathbf{p}\right) \tag{4.22}$$

For TemplateMatch we first compute the best match. If the ZSSD at the best match is below a threshold, the match is considered as a valid measurement $\mathbf{z} = \mathbf{p}^*$ otherwise the measurement is considered as failed and $\mathbf{z} = \mathit{Failure}$.

Note that $\mathbf{z}$ is to be used as the measurement of the projection of feature $\mathbf{y}_i$ later in the EKF update. Remember that $\mathbf{y}_i$ represents an idealised point and the template image describes the local neighbourhood of this point in the image. Thus, the template is to be constructed such that the projection of the feature point corresponds to pixel coordinates $\mathbf{0}$ in the template image.

Subpixel Refinement

The position obtained by exhaustive search as described above is accurate to integer pixel coordinates. For more accurate measurements, a successful match may be locally refined to subpixel accuracy.

Subpixel refinement is an optional step of the TemplateMatch procedure. The improvements achievable by this will be demonstrated later in this thesis. There are several methods for subpixel refinement but a detailed discussion is beyond the scope of this thesis. Specifically, in our implementation we employ iterative image alignment as described in the following.

In image alignment, the goal is to find the parameter vector $\widehat{\mathbf{p}}$ that minimises the objective function

$$\widehat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in X} \left(J\left(\mathbf{x}\right) - I\left(w\left(\mathbf{x};\mathbf{p}\right)\right)\right)^2 . \tag{4.23}$$

Here, $w\left(\,\cdot\,;\mathbf{p}\right) : \mathbb{R}^2 \to \mathbb{R}^2$ is a general *warp function* that is parameterised by a parameter vector $\mathbf{p}$. Starting from an initial estimate of $\mathbf{p}$, a local minimum is found using Gauss-Newton iterations. Specifically, we use the inverse compositional image alignment method of Baker et al. (2001).

We use the warp function $w\left(\mathbf{x};\mathbf{p}\right) = \mathbf{x} + \mathbf{p}$, i.e., the parameter vector is the image position of the template. To stay robust against intensity variations, we use the modified objective function

$$\widehat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_{\mathbf{x}\in X} \left(J\left(\mathbf{x}\right) - \bar{J} - I\left(w\left(\mathbf{x};\mathbf{p}\right)\right) + \bar{I}^*\right)^2 . \tag{4.24}$$

where $\bar{I}^*$ is the mean image intensity (4.21) computed at $\mathbf{p} = \mathbf{p}^*$. The initial guess for the parameter vector is the best match $\mathbf{p}^*$ found by the exhaustive template search.

The purpose of the iterative alignment is to refine the pixel-accurate match, thus the local minimum $\widehat{\mathbf{p}}$ should not be further than a pixel away from the initial guess $\mathbf{p}^*$. If it does, we assume that alignment converged to a spurious minimum and the subpixel refinement has failed. In this case, we have the choice to either keep the pixel-accurate template match or mark the measurement as failed, $\mathbf{z} = \mathit{Failure}$. If the refinement is successful we return $\mathbf{z} = \widehat{\mathbf{p}}$.

Feature Search in Stereo Images

We have described above how to locate a template in a monocular image. In our setting we employ a stereo camera, so the measurement includes disparity. Both the measurement $\mathbf{z}$ and the search region R are in (u, v, d) space. Instead of the input image I we have a stereo pair (I_r, I_l).

In this case, we proceed by projecting the search region into the right and left images to obtain monocular search regions R_r and R_l. Then we locate the template in the reference image, i.e., the right image, as described above. The resulting monocular measurement provides $\mathbf{z}_r = (u, v)^\top$.

It remains to find the disparity. We know that the match in the left image is must be located on the epipolar line v. Thus we reduce the left-image search region R_l to the pixels that lie on the epipolar line. Then we locate the template in the left image, as described above. The resultant monocular measurement provides $\mathbf{z}_l = (u', v')^\top$ and we obtain the disparity is obtained as $d = u' - u$.

Subpixel refinement is applied to both images separately. If any of the monocular measurements fail, the stereo measurement fails as well and we return $\mathbf{z} = \textit{Failure}$. Otherwise, we return the stereo measurement $\mathbf{z} = (u, v, d)^\top$.

4.7.2. Active Search

Active search is a top-down approach, using prior information to guide the measurement process. Prior information is used reduce the image area that has to be searched. The efficiency of the measurement process can be drastically increased by focussing processing resources in image areas where matches will be found with high probability. Also, smaller search regions reduce the chance of mismatches, because the requirement of global distinctiveness of a feature becomes one of distinctiveness within a small local neighbourhood.

Active search methodology is in contrast with the bottom-up approach often employed in off-line Structure-from-Motion (Hartley & Zisserman, 2000). Bottom-up approaches often assume unordered image sets; sequentiality is not exploited. All images are exhaustively searched for interest points and matches are established afterwards. Potentially the match for a point might be found anywhere in another image. Even with highly distinctive interest points the matching is ambiguous and prone to mismatches. Care must be taken to detect and remove such mismatches.

In visual SLAM, however, we have strong priors and a motion model which allows us, without even looking at the current image, to make rather accurate predictions about the measurements. Given the estimates of the camera pose and a feature point, we can compute the projection of the point in the current image. Taking uncertainty into account, we obtain a 2-dimensional probability distribution in the image space. Thresholding on the probability density leads to a tightly constrained elliptical search region around the predicted location. It would be highly unlikely to find the correct match outside this region (assuming that our model assumptions are correct).

More formally, let us assume that we are only measuring a single feature $\mathbf{y}_i$. The prediction of the measurement is $\mathbf{h}(\overline{\boldsymbol{\mu}}_t, \mathbf{0}) = \mathbf{y}_i^{\mathcal{I}}$, obtained by applying the measurement model to the current prior belief and the expectation of the noise vector, cf. Equations 4.13 and 4.16. The uncertainty of this prediction is given by the innovation covariance. By substituting the linearized measurement model into Equation 3.54 we obtain the belief about the measurement

$$\text{bel}(\mathbf{z}_t) = p(\mathbf{z}_t \,|\, \mathbf{z}_{1:t-1}) = \mathcal{N}(\mathbf{z}_t \,;\, \mathbf{y}_i^{\mathcal{I}}, \mathsf{S}) \tag{4.25}$$

where $\mathsf{S} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \overline{\boldsymbol{\Sigma}}_t \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^\top + \mathsf{R}_t$ is the innovation covariance.

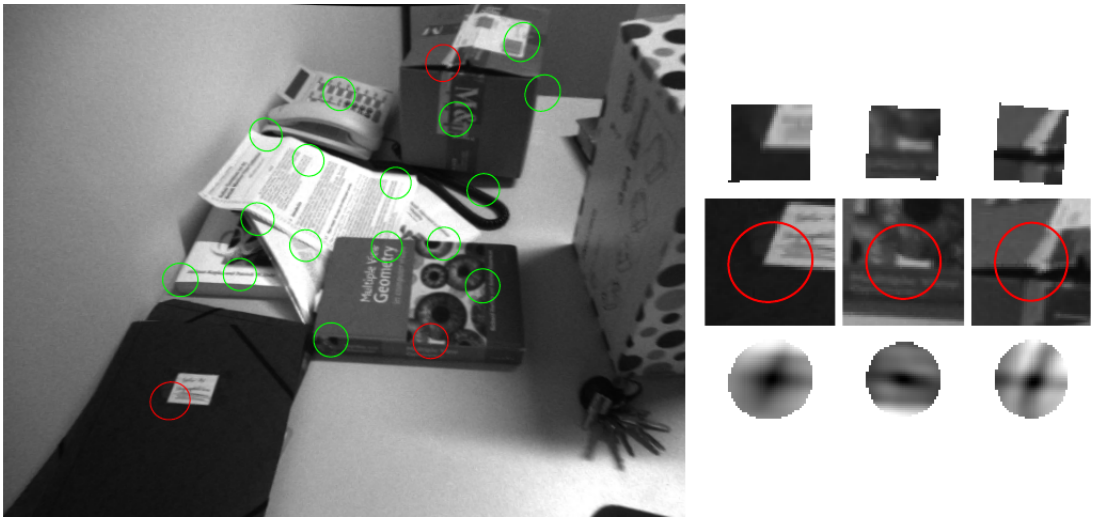


Figure 4.1.: Making measurements using active search. On the left, the input image is shown, with search ellipses overlaid. For the three ellipses outlined in red, exhaustive search is illustrated on the right. In the top row, the matching templates are shown. In the middle row, close-ups of the search regions are shown. In the bottom row, the ZSSD cost for the searched pixels are shown (darker pixels correspond to lower ZSSD).

As we are considering a single feature only, bel($\mathbf{z}$) is a 2-dimensional Gaussian distribution over the image coordinates of the features projection.[6] The Mahalanobis distance of an image point $\mathbf{z}$ to the predicted measurement is

$$D = (\mathbf{z} - \mathbf{y}_i^{\mathcal{I}})^\top \mathsf{S}^{-1} (\mathbf{z} - \mathbf{y}_i^{\mathcal{I}}). \tag{4.26}$$

We can restrict the template search to an elliptic image region[7] by thresholding on the Mahalanobis distance, where the threshold is selected as follows: The measurement $\mathbf{z}$ is Gaussian distributed. Thus, its Mahalanobis distance is χ^2 distributed with $d = 2$ degrees of freedom. For a given desired confidence level we find the Mahalanobis threshold from the inverse of the cumulative χ^2 distribution. For example, if we want ensure that the true measurement lies within the search region with probability 0.99 then we seek a threshold D_{max} such that

$$P(\chi_d^2 \leq D_{max}) = 0.99 \tag{4.27}$$

From the inverse cumulative distribution function we find $D_{max} = 9.21$. That is, with probability 0.99 the Mahalanobis distance to the true measurement is $D \leq 9.21$.

The method just described is used by our implementation of the PredictSearchRegion function to compute the image search region R for the current measurement candidate.

[6] For the stereo camera, it is a 3-dimensional distribution including disparity. This distribution can be projected to the left and right image of the stereo pair to obtain 2-dimensional distributions in both images.

[7] The iso-curves of the Mahalanobis distance are ellipses in the image.

An example of active search is given in Figure 4.1. It shows on the left a screen-shot from our SLAM system running on an indoor office-like scene. The 0.99 probability search ellipses for all currently visible features are overlayed. The image is 100 frames into the sequence. At this point the feature location estimates have already converged to quite accurate values. Thus, the predicted uncertainty is mainly due to the uncertain camera location. On the right of Figure 4.1, the template matching procedure is illustrated in detail for the three search ellipses outlined in red. In the top row, the matching template for each of the three features is shown, respectively. Close-ups of the search regions are shown in the middle row. The bottom row visualises the ZSSD cost for the pixels in the search region. Darker pixels indicate lower ZSSD cost, i.e., the darkest pixel in every region indicates the best match.

4.7.3. Selection Heuristic

We finish our description of the measurement process by looking at the measurement selection heuristic employed by SelectCandidate. Let us remember that the SelectCandidate function is responsible for determining the next feature we wish to measure. To make a decision it can use the following input: The prior state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ allows to make predictions about the image projections of features. The appearance information A may provide for example information on the range of suitable viewing angles and viewing distances for the features. Finally, the set of measurements made so far M is provided, which can be used to ensure measurements are spread well in the image.

The heuristic employed by SelectCandidate may vary strongly between different implementations. We give some general thoughts here, as well as details concerning our own implementation. We can structure the task into two steps. In the first step, suitable measurement candidates are selected according to predicted visibility and appearance. This step discards feature candidates which are currently behind the camera, which are seen from an inappropriate distance, etc. The second step concerns the selection among the remaining good candidates. We aim to process a live stream of camera images in real-time. There is only a limited budget of processing time available for each image. Often, the time budget is insufficient to measure all visible features.

Let us look at the two steps in more detail. Many of the features in the map can be discarded by some simple pre-selection rules. In our implementation, we check for the following. First of all, measurement candidates should be predicted to be visible, i.e., they should be in front of the camera, and their projection should be within the image boundary. Secondly, the predicted camera–feature distance is limited to a range reasonably similar to the camera–feature distance when the feature was initialised. The reason for this is that when the camera moves much further away from a feature than upon initialisation, then the search template will become very small (eventually the size of a pixel or less). Clearly, template matching will not provide a reliable measurement in this case. Similar considerations hold for moving too close to a feature. Finally, the viewing angle can be limited to be close to the initial viewing angle. Increasingly different viewing angles cause increasing distortions in the search template until finally the feature becomes invisible because the camera is looking at its back. If the surface normal of the feature patch is available, this can be also used to restrict the viewing angle.

After these (or similar) sanity checks, a number of candidate measurements usually remain. There may not be enough time to carry out the feature searches for all of them. We want to select those measurements that yield maximum information, i.e., the largest reduction of uncertainty. Intuitively, the innovation covariance, i.e., the size of the search region, is a measure for the information expected of a measurement. If the search region is small then it is already very well known where the projection of the feature is going to be located – the "surprise value" of that measurement is low. If the search region is large then the measurement provides more information. Thus, one heuristic is to measure those features with the highest innovation covariance.[8] This is the strategy used in (Davison et al., 2007). Another possibility is to select measurements that are well-spread across the image because this constrains the camera pose well.

Our system uses a combination of the heuristics mentioned above: The reference image of the stereo pair is divided into a 4×4 grid. Measurement candidates are sorted into the grid according to their predicted image positions. Within each grid cell, candidates are ordered by the volume of their uncertainty region. In each grid cell measurements are attempted starting with the feature with the largest uncertainty region. Our SelectCandidate implementation attempts feature measurements until one successful measurement per grid cell has been made or all candidates in this cell have been attempted. Then, the measurement loop is terminated by returning the *None* candidate.

4.8. Map Management

The final step in the VisualSlam loop is the ReviseMap function which is responsible for *map management*, i.e., dynamically adding and removing features to ensure that the map contains at all times sufficient high-quality features for stable operation. In practice, new features are initialised and added to the map when new parts of the environment are explored. Occasionally, features are deleted from the map, for example if it turns out that attempted measurements keep failing. ReviseMap is presented in Algorithm 14. The algorithm comprises two major parts, namely the removal of bad features and the initialisation of new features.

4.8.1. Removing Features

The first part in lines 1–3 of Algorithm 14 concerns removing bad features. A feature is considered "bad" if it cannot be measured reliably or violates our assumptions about the scene. A feature can be difficult to measure for instance if it is frequently occluded or lies on an area with repetitive texture. A feature can violate assumptions for instance if it is located on a moving object. Also, a feature can be spurious for instance if it lies on an occlusion boundary.

[8] However, simply ordering candidate measurements by decreasing innovation covariance neglects inter-measurement correlations. Recently a theoretically well-founded solution was proposed by Chli & Davison (2008). Their "active matching" tightly integrates measurement selection with active search, taking full account of correlations. In the process they are also able to resolve ambiguous matches (a topic which is beyond the scope of this overview).

Algorithm 14: ReviseMap

Input: State estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, Current image I, Auxiliary state A, S
Output: Revised state estimate $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, Revised auxiliary state A, S

```
   // remove bad features
1  B := SelectBadFeatures (S)
2  μ, Σ := RemoveFromState (μ, Σ, B)
3  A, S := RemoveFromAux (A, S, B)

   // initialise new features
4  if InitialisationRequired (μ, Σ) then
5  |  C := FindCandidates (I)
6  |  F := SelectNewFeatures (μ, Σ, C)
7  |  μ, Σ := AddToState (μ, Σ, F)
8  |  A, S := AddToAux (A, S, F, I)
9  end
10 return μ, Σ, A, S
```

Bad features can often be detected using simple measurement statistics. This is done in line 1. The SelectBadFeatures function takes the measurement statistics S as input and uses a heuristic to label current map features as bad. The function returns a list B of indices of bad features. In our implementation, a feature is considered bad if more than 50% of the attempted measurements failed. A feature is also considered bad if more than 10% of the measurements did not pass the joint compatibility test.

Features listed in B are removed from the probabilistic state vector in line 2 using the RemoveFromState function. Removing a feature is a very simple operation: To delete a feature from the map we marginalise the joint belief over the state of the feature. Lets assume we want to remove $\mathbf{y}_i$. Then we need to marginalise

$$\text{bel}(\mathbf{x}_v, \mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_n) = \int \text{bel}(\mathbf{x}_v, \mathbf{y}_1, \ldots, \mathbf{y}_n)\, d\mathbf{y}_i. \tag{4.28}$$

In the Gaussian state representation, this amounts to simply deleting rows and columns corresponding to $\mathbf{y}_i$ from the mean vector and covariance matrix. This is precisely what RemoveFromState does for the features listed in B.

The corresponding entries from the auxiliary state are deleted by RemoveFromAux which is called in line 3. If feature $\mathbf{y}_i$ is removed then the corresponding appearance a_i and the measurement statistics s_i are removed from A and S respectively.

4.8.2. Adding Features

The second part of Algorithm 14 in lines 4–9 handles the initialisation of new features.

Initialisation does not occur in every frame. The function InitialisationRequired uses a heuristic to decide whether new features should be initialised in the current frame.

Its input is the probabilistic state estimate. This can be used to get the visibility and image positions of the features currently in the map. Based on this information, the InitialisationRequired heuristic decides whether new features should be initialised. One goal of the heuristic is to ensure that the current map is sufficient for tracking during the next frame. This requires that enough features will be visible during the next frame. To minimize the risk of tracking failure new features should be frequently initialised when old features move out of the image. On the other hand, adding features too often increases the map size and decreases the size of the environment that can be covered in real-time operation. Thus, features should be sparsely initialised. The specifics of the heuristic vary based on the relative importance put on these conflicting goals. For instance, Davison et al. (2007) require a specific number of features (they use 12) to be visible in every image. The heuristic used in our system is tailored to our feature representation and will be described later, in Section 6.6.3.

If InitialisationRequired returns $\top$ the *if*-block in lines 5–8 is executed, i.e., we try to initialise new features in the current frame. We proceed as follows.

First, candidate features are identified in the current image using FindCandidates. This function returns C, a set of *feature candidates*. Each candidate is a tuple $(\mathbf{z}, q)$ comprising image position of the feature candidate, $\mathbf{z}$, and the feature quality q. The image position is $\mathbf{z} \in \mathbb{R}^2$ in the monocular case. In the stereo case, $\mathbf{z} \in \mathbb{R}^3$ includes the disparity. The scalar q is a measure for the expected quality of the feature candidate. For instance, q can be a saliency score of the image neighbourhood of the candidate.

The details of FindCandidates vary between different implementations. Usually, FindCandidates will employ some saliency detector to extract promising candidates from the image. Our implementation consists of the following steps. First we run the FAST detector (Rosten & Drummond, 2006) with non-maxima suppression to detect corners in the reference image. FAST provides no measure of corner quality. Therefore, for each of the detected corners we compute the Shi-Tomasi score (Shi & Tomasi, 1994). This provides a measure of local saliency that is used as the quality q of the feature. Finally, a match to a small neighbourhood of the corner, i.e., 7×7 pixels, is searched for in the left image to provide the disparity. The match is found by exhaustive search along the epipolar line for the pixel window that minimises the SSD. If the minimum SSD is below a threshold, the candidate is accepted and added to C.

The set C provided by FindCandidates often contains many more feature candidates than we require. The function SelectNewFeatures chooses from the set of candidates C a subset F that will actually become new map features. Again, a heuristic is used to make this choice. In the following we describe our implementation. New feature candidates are selected such that they have a high saliency score q and are distributed in the image. These two criteria are combined in a score function that is computed for every feature

$$s = \lambda q + D_{min}^2. \tag{4.29}$$

Here, q is the saliency score and D_{min} is the image distance to the closest existing feature. Existing features refers to features from the map, as well as other candidates that have already been selected into F. The tuning parameter λ weights the relative importance of the criteria. The candidate with the best score is selected and added to F. Then the scores

of the remaining candidates are re-evaluated.[9] The process is repeated until a predefined number of candidates have been selected or until the score for all remaining candidates is below a threshold.

Then function AddToState initialises the new features into the probabilistic state. This is accomplished by inserting new blocks into the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The new covariance blocks need to correctly reflect the correlations between the existing state and the new feature. The details of this computation will be explained shortly.

Finally, the auxiliary state is extended by AddToAux in line 8 of Algorithm 14. Appearance information a_i for every new feature i is extracted from the input image I. In our system this comprises a feature template that is cropped from the image. Also, an empty measurement statistics s_i is created for every new feature.

In the remainder of this section we will look at the initialisation of features into the probabilistic state. In the following we derive the required manipulations of $\boldsymbol{\mu}, \boldsymbol{\Sigma}$. Suppose that we make an observation $\mathbf{z}$ of a new feature, i.e., a candidate feature has been selected at $\mathbf{z}$ in the current image. We want to add this newly observed feature to the state vector. Suppose there are n features currently in the state vector and the new one becomes $\mathbf{y}_{n+1}$. We assume that $\mathbf{z}$ is generated by a measurement model, similar to the one discussed in Section 4.6. We assume that $\mathbf{z}$ is a noisy version of a hypothetical, perfect measurement $\mathbf{y}^{\mathcal{I}}_{n+1}$, i.e., the measurement made in the best circumstances. The measurement noise is assumed zero-mean Gaussian with covariance R. Like in the measurement model, $\mathbf{y}^{\mathcal{I}}_{n+1}$ denotes the ideal projection of the new feature $\mathbf{y}_{n+1}$ in the current image. Given the measurement, we can write its belief as

$$\mathrm{bel}(\mathbf{y}^{\mathcal{I}}_{n+1}) = p(\mathbf{y}^{\mathcal{I}}_{n+1} \,|\, \mathbf{z}) = \mathcal{N}(\mathbf{y}^{\mathcal{I}}_{n+1}\,;\, \mathbf{z}, \mathsf{R}). \tag{4.30}$$

Note that the measurement $\mathbf{y}^{\mathcal{I}}_{n+1}$ is completely uncorrelated to the current state. Nothing we know about the current camera pose or any other feature can tell us anything about this new measurement. Therefore, the joint belief about the measurement and the state is simply the product of the two distributions.

$$\mathrm{bel}(\mathbf{x}, \mathbf{y}^{\mathcal{I}}_{n+1}) = \mathrm{bel}(\mathbf{x}) \cdot \mathrm{bel}(\mathbf{y}^{\mathcal{I}}_{n+1}) \tag{4.31}$$

This is the product of two Gaussians which we can write as a single Gaussian in the augmented state vector

$$\mathbf{x}_{aug} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y}^{\mathcal{I}}_{n+1} \end{pmatrix}, \tag{4.32}$$

$$\mathrm{bel}(\mathbf{x}_{aug}) = \mathcal{N}(\mathbf{x}_{aug}\,;\, \boldsymbol{\mu}_{aug}, \boldsymbol{\Sigma}_{aug}) \quad \text{with} \quad \boldsymbol{\mu}_{aug} = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{z} \end{pmatrix}, \;\; \boldsymbol{\Sigma}_{aug} = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathsf{R} \end{bmatrix}. \tag{4.33}$$

We would like to extend the state $\mathbf{x}$ by the new feature $\mathbf{y}_{n+1}$, and compute the belief for this new state. For this we need an *inverse measurement model* $\mathbf{g}$ which computes the new features position given the current camera pose and the initial measurement.

$$\mathbf{y}_{n+1} = \mathbf{g}(\mathbf{y}^{\mathcal{I}}_{n+1}, \mathbf{x}_v) \tag{4.34}$$

[9] Their D_{min} score might have decreased because the newly selected candidate is now their closest existing feature.

Given such a model, we can write the new state as a function of the augmented state

$$\mathbf{x}_{new} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y}_{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{g}(\mathbf{y}^{\mathcal{I}}_{n+1}, \mathbf{x}_v) \end{pmatrix} \tag{4.35}$$

We linearise this function and project the Gaussian $\mathrm{bel}(\mathbf{x}_{aug})$ through the linearisation to obtain the new belief with feature $\mathbf{y}_{n+1}$ added

$$\mathrm{bel}(\mathbf{x}_{new}) = \mathcal{N}(\mathbf{x}_{new}\,;\, \boldsymbol{\mu}_{new}, \boldsymbol{\Sigma}_{new}) \tag{4.36}$$

with

$$\boldsymbol{\mu}_{new} = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{g}(\mathbf{z}, \boldsymbol{\mu}_{\mathbf{x}_v}) \end{pmatrix}, \quad \boldsymbol{\Sigma}_{new} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma}\frac{\partial \mathbf{g}}{\partial \mathbf{x}}^{\top} \\ \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\boldsymbol{\Sigma} & \frac{\partial \mathbf{g}}{\partial \mathbf{x}}\boldsymbol{\Sigma}\frac{\partial \mathbf{g}}{\partial \mathbf{x}}^{\top} + \frac{\partial \mathbf{g}}{\partial \mathbf{y}^{\mathcal{I}}_{n+1}}\mathsf{R}\frac{\partial \mathbf{g}}{\partial \mathbf{y}^{\mathcal{I}}_{n+1}}^{\top} \end{bmatrix}. \tag{4.37}$$

This gives the required new mean and covariance blocks. In the AddToState function, the above operation is repeated for every new feature.

It remains to define the inverse measurement model $\mathbf{g}$. For a stereo camera, this can simply be obtained by inverting Equation (4.16).

$$\mathbf{y}_{n+1} = \mathbf{g}(\mathbf{y}^{\mathcal{I}}_{n+1}, \mathbf{x}_v) = p^{\mathcal{WC}}(\pi_s^{-1}(\mathbf{y}^{\mathcal{I}}_{n+1}); \mathbf{x}_v). \tag{4.38}$$

For a monocular camera, the situation is more difficult. A monocular image measurement back-projects to a full ray of possible 3D points in the scene. In contrast to the stereo projection function, the monocular projection function is not invertible. There is no equivalent to Equation (4.38) in this setting, because generally it is not possible to reconstruct the 3D feature position from a single measurement. Approaches to deal with this include using a special initialisation stage for new features (Davison, 2003; Solà et al., 2005) and the use of more suitable feature parameterisations (Montiel et al., 2006).

4.9. Parameter Settings

In the discussion of our visual SLAM algorithm the settings of various parameters have been left unspecified. In this section, we provide the settings that were used for experimentation throughout this thesis.

All real-world sequences have been recorded using a Point Grey Bumblebee® stereo camera, depicted in Figure 4.2. The Bumblebee has a baseline of 11 cm. A stereo pair consists of two 640×480 images. Each eye has a horizontal field of view of 65°, which corresponds to a focal length of 505 pixels. We also perform simulation experiments on rendered image sequences. The parameters of the renderer are set up to resemble the Bumblebee, as well.

The Bumblebee operates at a frame-rate of 30Hz which means that the per-frame time budget is 33 ms. The raw images provided by the camera are Bayer-coded color images and exhibit radial distortion. Before presenting the images to the SLAM system they are Bayer decoded and rectified. This is performed efficiently in parallel on a GPU. Together, image

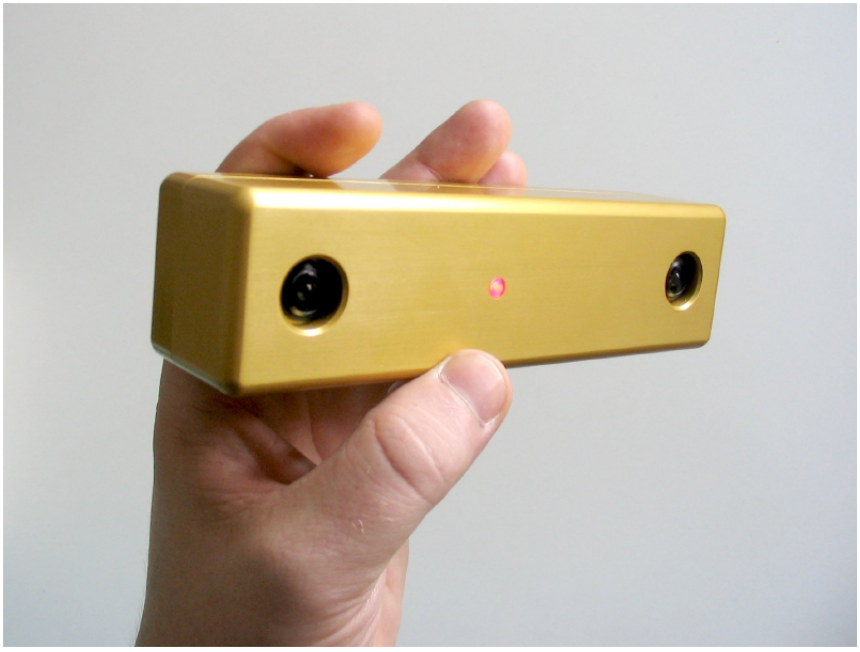


Figure 4.2.: Point Grey Bumblebee® stereo camera.

acquisition, decoding, and rectification take approximately 3.5 ms out of the processing budget.

For the recorded test sequences the camera is held in hand and freely moved in the environment. The noise settings of the constant-velocity model have been empirically tuned, such that fast but smooth hand-held motions are robustly tracked. In particular we assume a standard deviation of $7\,\mathrm{m}\cdot\mathrm{s}^{-2}$ for the linear acceleration components, and a standard deviation of $12\,\mathrm{rad}\cdot\mathrm{s}^{-2}$ for the angular acceleration components. We assume that individual components are uncorrelated. Thus, we use the diagonal process noise covariance

$$\mathsf{Q} = \begin{bmatrix} 49 & & & & & & \\ & 49 & & & & & \\ & & 49 & & & & \\ & & & 144 & & & \\ & & & & 144 & & \\ & & & & & 144 & \end{bmatrix}. \tag{4.39}$$

Point feature appearance is stored as 21×21 template selected from the initial camera image. For matching, we warp the feature template to account for the predicted changed camera viewpoint. The shape and size of the matching template is determined by this warping operation. We experiment with both pixel accurate and subpixel accurate template matching. To further speed up template matching, we optionally restrict the template search to image pixels which have at least one FAST corner in their 8-neighbourhood.[10] Regardless of the matching procedure, we assume that 2D image mea-

[10] Computation of FAST corners does not introduce any overhead because the FAST detector has to be

surements can be made with 1 pixel standard error in u and v direction. We further assume that the u errors are uncorrelated between left and right image, which results in a standard deviation of $\sqrt{2}$ pixel for the disparity. We assume that individual error components are uncorrelated. Thus, we use the diagonal measurement noise covariance

$$\mathsf{R}_i = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 2 \end{bmatrix} \tag{4.40}$$

for every feature measurement.

The above setting are used for all experiments presented in this thesis.

run on the input images anyway to determine candidate locations for new features.

5
Evaluating Visual SLAM

5.1. Introduction

Building systems for camera-based simultaneous localisation and mapping is a challenging task. A complete implementation of visual SLAM as described in Chapter 4 has to deal with a wide range of issues from low-level image processing to high-level decision-making about when and where to remove old or select new map features. Real-time constraints and noisy input data raise further difficulties.

Given the complex nature of the implementations and the inherent dependency of the input data, the performance analysis of such systems is considerably hard as well. This is still largely an unsolved problem. There is a lack of generally accepted performance measures, test frameworks, and ground truth benchmark problems. Most researchers test by visually inspecting their systems on recorded image sequences, or measuring accuracy on simulated data of simplified point-cloud-like environments. Both approaches have drawbacks. Recorded sequences lack ground truth. Point cloud simulations tend to oversimplify low-level aspects of the problem. For instance, they abstract from the image processing layer which has a great influence on the overall system performance.

In this chapter, we present the SLAMDUNK evaluation framework which allows to analyse the performance of visual SLAM systems on rendered image sequences. The intention is to move simulation towards more realistic conditions while still having ground truth. In other areas of computer vision, e.g., optical flow estimation, rendered images are now part of the standard benchmarks (Scharstein & Szeliski, 2009). We believe that the field of visual SLAM can similarly benefit from this approach. High-quality rendered images are close enough to real-world images to violate the assumptions usually made in the image processing part of visual SLAM. Many effects occurring in real sequences can be emulated in rendered images, like non-Lambertian surfaces, changing illumination, motion blur, shading, and sensor noise. This allows for a deeper study and evaluation of the whole visual SLAM system from the lowest to the highest level. With SLAMDUNK, a complete

and extensible framework is provided which addresses all aspects, from rendering to ground truth generation and automated evaluation.

A novel aspect of the SLAMDUNK framework is *adaptive* generation of ground truth. The framework does not interfere with the visual SLAM system under evaluation in any way. The input to the SLAM system is an image sequence and the system is responsible for selecting and tracking features. The ground truth map is generated afterwards by the evaluation framework according to the features that were selected by the SLAM system. From this ground truth, we can also compute the ideal measurements, a reference against which the results of feature matching may be compared. Additionally, SLAMDUNK offers support for arbitrarily complex scenes and trajectories, different camera and image degradation models, automated rendering and evaluation – all in one integrated and extensible framework. A main goal in designing the framework was to make it easy to add new scenes and evaluation code, such that it can be readily used and extended by others for their own research. The framework was developed in joint work with Jan Funke. It is completely built on free software and is publicly available (Funke & Pietzsch, 2009b) under the GNU Public License.

The framework may be used to obtain answers about single components of a visual SLAM system, e.g., "What is the influence of different feature removal heuristics to trajectory and map accuracy?", "Which feature matching method minimizes the measurement errors?". In addition, the availability of the ground truth may be used for quick hypothesis checking, e.g., "What would be the benefit of accurate estimation of feature normals?"

The remainder of this chapter is structured as follows. In the next section we review related work. In Section 5.3 we give a high-level overview of the framework and its components. These components are detailed in Sections 5.4–5.7. This is followed by some experiments in Section 5.8 which exemplify use cases of the framework. In particular, we assess the benefit of feature normal estimation and subpixel-accurate matching and analyse the influence of motion blur. We conclude with a discussion and open problems for future work in Section 5.9.

5.2. Related Work

We can broadly distinguish four kinds of approaches to experimentally evaluate visual SLAM systems. In the first class of approaches, the system is run on recorded image sequences where ground truth is not available. Performance analysis is based on *visual inspection*. Almost all recent work include an evaluation of this kind (e.g., Klein & Murray, 2008; Eade & Drummond, 2008; Chekhlov et al., 2007; Civera et al., 2007; Williams et al., 2007; Clemente et al., 2007; Eade & Drummond, 2006b). Typically, these papers show screenshots of the system map/trajectory visualisation output, verbally explain what is going on at prototypical images in the sequence, and give interpretations of the visual results. The overall performance is usually described in broad categories such as whether a sequence is "well-tracked throughout" or "fails after x seconds", and whether loops are closed successfully. Using this approach, it is possible to illustrate strengths and weaknesses of a particular technique in the face of all the problems and artifacts occurring on real-world examples. However, it does not provide any hard quantitative results. Given

the complex structure of the implementations, comparisons between different systems based on visual inspection alone are difficult. Usually, a lot of heuristics are involved in the implementation. It is often impossible to say what particular combination of components is responsible for, e.g., a successful or failed loop closure.

To obtain a quantitative measure of achieved reconstruction accuracy, some authors use *geometric constraints* in the scene. For instance, when the scene is known to be planar, the distance of estimated features to the plane (which is either known *a priori* (Smith et al., 2006) or an estimated best fit (Eade & Drummond, 2008)) can be evaluated. Other constraints are right angles known *a priori* (Smith et al., 2006) and hand-measured distances in the scene (Solà et al., 2007). In the latter case, inaccuracies cannot be avoided, of course. This second kind of approaches complements visual inspection by giving quantitative results at least for some restricted scene types.

Another solution is to record measurements made by the visual SLAM system and compute a *maximum likelihood reconstruction* using bundle adjustment (Triggs et al., 1999). The SLAM estimate can then be compared to this solution (e.g., Eade & Drummond, 2007). This is well-suited for evaluating the performance of the filtering algorithm, for instance the influence of model linearisation. However, it abstracts from other influences such as feature matching accuracy, data association, and feature selection heuristics, that is, anything leading up to the 2D measurements.

Finally, SLAM systems can be run on *simulated data*, and evaluated against ground truth (which is of course available in simulations). Abstraction from systematic influences at the image processing level can be a problem here, too. Often simulated scenes are modelled as point clouds and measurements obtained as projections of these points with additive Gaussian noise (e.g., Civera et al., 2007; Bekris et al., 2006; Chiuso et al., 2002). Besides making idealised assumptions about feature matching, another drawback in (Chiuso et al., 2002) is that map features are predefined, i.e., the SLAM system has no control of when and where to initialise features.

We propose to use rendered image sequences to improve upon point-cloud simulation. In two recent papers, visual SLAM results were evaluated on rendered images. Chekhlov et al. (2007) use a camera trajectory in front of a texture-mapped plane to analyse the performance of their exemplar-based feature descriptor. Klein & Murray (2007) test their system on a synthetic scene produced in a 3D rendering package. They compare their results and those of an EKF-based system with respect to ground truth. However, this comparison is limited to the estimated trajectories. The estimated maps are only compared by visual inspection.

In contrast to these authors, we generate ground truth feature positions for the rendered sequences. From this ground truth, we can also compute the ideal measurements, a reference against which the results of feature matching may be compared.

5.3. Framework Overview

The motivation for building the SLAMDUNK evaluation framework was the possibility to easily set up and perform visual SLAM experiments on rendered image sequences. As we are convinced that such a framework is of interest to other researchers as well, we focused

Figure 5.1.: POV-Ray rendering of the office scene by Jaime Vives Piqueres.

on the following requirements. The framework should be freely available, that means, built on free software only. It should be easy to use, that is, its adaptation to existing SLAM implementations should be straightforward. Moreover, the framework should be extensible to encourage contributions from the community.

For image generation, we use the POV-Ray command line ray tracer (Persistence of Vision Pty. Ltd., 2004), which is free software and allows rendering of realistic images. POV-Ray features a powerful scene description language (SDL). The SDL provides a trace feature which calculates the coordinates of the intersection of a ray with the nearest scene object. This feature is crucial for adaptive generation of ground truth, as described in Section 5.7.1. Moreover, a range of high detail scenes and object models is freely available for this renderer. For example, Figure 5.1 shows a rendering of the "office" scene by Jaime Vives Piqueres.[1]

A large part of our framework deals with automatizing the processes of creating image sequences and evaluating experiment results. For the interface between the evaluation framework and the visual SLAM system, we provide a lightweight API in C++ that allows access to the created sequence images and straightforward data logging of the SLAM process.

For system integration and easy extendability, all relevant data throughout the framework is stored in an XML format. This includes camera and trajectory definitions, log data, ground truth, and intermediate evaluation results. The log data is processed using an extensible set of Python scripts.

[1]The source code for the scene is available from his website `www.ignorancia.org`.

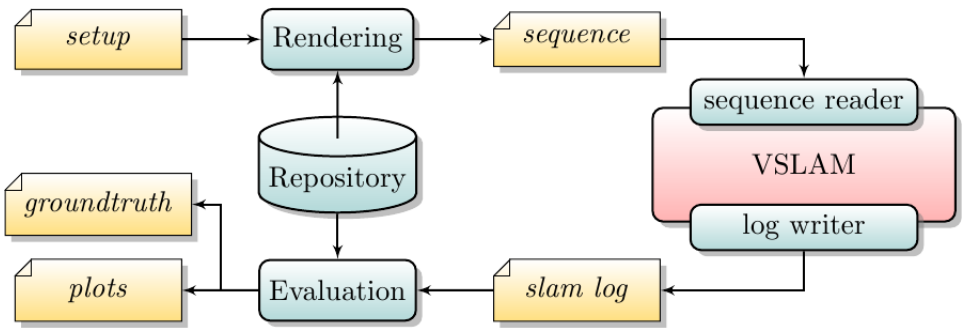


Figure 5.2.: Overview of the SLAMDUNK evaluation framework. An image sequence is rendered and provided as input to the visual SLAM system under evaluation. A log of the systems operation is created and processed afterwards by the *evaluation system*. Together with the exactly known scene structure and trajectory, the log is used to generate ground truth for feature positions and measurements. The final result is a number of error plots comparing logged estimated values and ground truth. Interface classes *sequence reader* and *log writer* are provided to allow easy adaption to existing systems.

Our evaluation framework consists of four components: a *repository*, a *rendering system*, an *interface* for visual SLAM systems and an *evaluation system* (see Figure 5.2). The *repository* is used to store common items such as the descriptions of 3D scenes, camera trajectories and camera definitions which are needed by both the *rendering system* and the *evaluation system*. The *rendering system* is used to create image sequences from a specific setup of *repository* items. Using the *interface* (the lightweight C++-API), each of these sequences can be used instead of a regular camera (mono or stereo) to perform experiments on the SLAM system under consideration. This *interface* can also be used to easily create structured log data during the experiment which are required for the *evaluation system* afterwards. For example, the log data includes estimated location of camera and/or map features, 2D image measurements, and feature initialisation events. The *evaluation system* consists of an extensible set of Python scripts that perform various evaluation tasks, e.g., the computation of ground truth and the creation of data plots for the experiments.

Communication between these components is done by exchanging XML files which we found to be the most convenient solution. Actually, a large part of the framework targets on making it comfortable to read and write these XML files, so that it should be easy for other researchers to contribute new sequences, camera models and evaluation scripts.

The SLAMDUNK framework is completely written in Python except for the *interface*, which is written in C++.

5.4. Repository

The repository consists of building blocks which can be assembled into different combinations to create image sequences. The repository contains items of the following types: *scene description*, *camera configuration*, *trajectory*, and *rendering options*.

A *scene description* defines the 3D environment in which the camera will be placed. We use the POV-Ray (Persistence of Vision Pty. Ltd., 2004) raytracer for image generation, and so the *scene description* is in the native POV-Ray format.

All other repository items are stored as simple XML files. A *camera configuration* consists of camera type (mono/stereo), internal calibration parameters, image resolution, shutter speed, and frame rate. A *trajectory* describes camera motion as a list of keyframes, where every keyframe comprises a timestamp and a camera pose. *Rendering options* influence the quality of the ray-traced images. For instance, there are options to control the quality of anti-aliasing, lighting, and motion blur calculation.

5.5. Rendering System

The input to the rendering system is a *setup* description. Every *setup* is used to generate one *sequence*.

A *setup* selects a combination of a *scene description*, a *camera configuration*, a camera *trajectory*, and a set of *rendering options* from the *repository*. Using the setup information, the camera is moved along the trajectory within the scene and an image sequence is rendered. Begin and end of the sequence are defined by the timestamps of the first and last trajectory keyframes. The number of frames to be rendered in-between depends on the framerate of the camera. In general, trajectory keyframes need not match the framerate of the camera or even be equidistant in time. Camera poses for frames between keyframes are interpolated between keyframe poses. Camera positions are linearly interpolated, spherical linear interpolation is used for the camera rotation. Being an advantage of this approach, the trajectories can be obtained from a variety of sources, e.g., hand-crafted using some modeling program, recorded from an IMU, and so on. However, keyframes should not be too far apart in time, because otherwise the linear interpolation may cause discontinuities. Currently, the framework includes some synthetic trajectories and trajectories reconstructed from real image sequences.

After completing the setup step, the camera is placed at each interpolated frame pose, internal camera parameters are set as specified in the *camera configuration*, and an image is rendered using POV-Ray (two images in the stereo case). Motion blur effects are simulated by rendering and averaging multiple images from nearby poses. The amount of blur is controlled by the shutter speed, which defines the range of poses around the timestamp of the current frame. In addition to motion blur, the only image degradation effect currently available is additive Gaussian noise. We have decided to add the noise on-the-fly in the *sequence reader* of the VSLAM API (see Section 5.6) because this allows to test the same setup with different noise levels without re-rendering the sequence again and again.

As the result of the steps described above, we obtain a set of images along the trajectory. In the case of a stereo camera, the left and right image of every stereo image pair are merged into a single output image. Finally, a *sequence* description file is created. A *sequence* consist of a list of timestamped images and a set of parameters that should be available to the visual SLAM system (e.g., camera calibration settings). The images and parameters are accessible through the interface described in the next section.

5.6. VSLAM Interface

Interfacing the evaluation framework with existing visual SLAM systems is made easy using a lightweight API consisting of two parts, the *sequence reader*, and the *log writer*. We chose to provide the API as a C++ library because many existing systems are implemented in this language (e.g., Davison, 2003; Eade & Drummond, 2007; Klein & Murray, 2007). A reason for this may be that C++ allows to write fast programs which is important because implementations often have a focus on real-time operation. Moreover, excellent linear algebra and image processing libraries are available.

5.6.1. Sequence Reader

The *sequence reader* provides camera parameters and images of a *sequence*. It is supposed to replace the camera driver of the SLAM system. The *sequence reader* is provided with an *experiment* description in XML, containing the name of the sequence to process.

Additionally, an *experiment* may contain arbitrary parameter settings which can be queried as key-value pairs. This should allow for a large number of experiments to be run in a batch (possibly in parallel). One possible scenario is to run a SLAM system on a number of sequences with varying parameter settings and observing the influence on the overall result. Another attractive option is to compare the results of different systems on a benchmark set of sequences, identifying the advantages and drawbacks of the different approaches.

5.6.2. Log Writer

The *log writer* collects data from the SLAM run and writes them into a set of *slam log* files. In particular, the *slam log* consists of the following data items (for every frame of the sequence):

- the estimated camera pose,
- the estimated map, i.e., all 3D feature positions,
- the measured 2D feature positions, and
- the 2D positions of newly initialised features.

Please note that the evaluation methods and measures currently implemented are targeted at point-feature-based systems, i.e., systems that maintain a representation of the map as a set of 3D point features. Every feature point is identified by a unique feature-ID, which appears in every measurement and position estimate.

Logging the initial image locations of new features is important for two reasons: Firstly, this data is used to obtain ground truth 3D positions for all features. Secondly, this data can be also be fed back into the *sequence reader* in another experiment. This way, we can force the visual SLAM system to initialise the same features in every run, alleviating the comparison of the resulting maps.

Currently, the log does not include uncertainty information, e.g., covariance matrices. Covariance information might not be available directly (for instance in a system using submaps, or a particle filter). If it is available, it might not be in the right parameter space (for instance in systems using inverse depth parameterisations). However, to compare different systems, uncertainty information should be represented in a unified way. This necessitates covariance projection and/or approximation prior to logging. Addressing the effects of such approximations correctly in evaluation measures is not straightforward. A general solution seems rather difficult here. For specific scenarios it is quite easy to extend the log formats to incorporate uncertainty information.[2]

5.7. Evaluation System

The *evaluation system* consists of an extensible collection of *evaluation scripts*. Some scripts compute the ground truth required for the evaluation. Other scripts are used to analyse the outcome of the experiments, as described below.

In the given framework, both types of scripts are handled in the same way: Each script may depend on certain files, e.g., *slam log* files, and produce one ore more files, e.g., plot data or ground truth. The output of one script may be the input to another, which results in a dependency graph for running these scripts. The framework takes care of resolving these dependencies, which allows easy integration of new scripts by stating their requirements and outcomes.

5.7.1. Adaptive Generation of Ground Truth

Ground truth is determined as an intermediate step in the *evaluation system* by dedicated *evaluation scripts*. Ground truth is computed for each set of *slam log* files. The *slam log* comprises a copy of the *setup* use to create the sequence. Thus, the *evaluation scripts* can refer to the correct repository items for ground truth generation. Ground truth data consists of the real trajectory, the map of real feature positions, and the set of ideal measurements.

The real trajectory is readily available because it was used to render the sequence. The ground truth map is generated for the feature points that were selected by the SLAM system. From the *slam log*, the timestamp of each feature initialisation is known, as well as the pixel coordinates the feature was initialised at. Using the initialisation timestamp, the camera pose for each feature initialisation can be determined straightforwardly from the ground truth trajectory. Next, we use POV-Ray's trace facility to compute the 3D position of the feature: We place the camera at the initialisation pose and cast a ray from the camera center through the features initial pixel coordinates on the image plane. POV-Ray computes the nearest intersection of this ray with the scene. This intersection yields the ground truth feature position. Crucially, the ground truth map is generated according to the features selected by the SLAM system. This means that there may be different ground truth maps for different runs with the same sequence. This adaptive approach

[2] We use this possibility later in this thesis. In Section 7.9, we carry out experiments where we log the covariance of the estimated camera pose.

provides more flexibility than point cloud simulations. In a point cloud simulation the evaluation system dictates which features the SLAM system should initialise.

Given the ground truth map and trajectory, the projection of every feature point in every frame can be computed. These projections are the *ideal measurements*, i.e., the best image measurements that feature matching could provide. Access to the ideal measurements enables the evaluation of image measurement error, which is another advantage over point cloud simulation.

5.7.2. Evaluation Procedures

The other type of scripts use ground truth and estimated data to compute quantitative error measures. Currently, we provide basic evaluation measures for trajectory error, map error, and measurement error.

The trajectory error is evaluated with respect to translation and rotation. The *absolute camera position error* is computed as the Euclidean distance between the ground truth and estimated camera translation. To evaluate the *absolute camera rotation error* we compute the rotation that transforms the estimated camera orientation into the ground truth camera orientation. The absolute camera rotation error is computed as the angle of this differential rotation in degrees. As another measure of rotational error, we provide the Frobenius norm of the difference of estimated and ground truth camera rotation matrices. These errors are computed for every frame of the sequence. The result is plot data of translation error over time, and rotation error over time, respectively.

The *map error* for every frame is computed as the root mean squared difference of estimated and ground truth map features. The result is plot data of map error over time. To compute the *best-fit map error*, the estimated map is rigidly transformed to minimize the mean squared difference of estimated and ground truth map. Then we compute the map error between the transformed estimated map and the ground truth. Optionally, the best-fit transformation can include scaling. The latter is especially interesting for monocular SLAM as it does not penalise unobservable scale.

For each measurement, the image distance in pixels between the ideal and actually measured image projection is computed. The result can be used to generate scatter plots of measurements over all or individual features.

All these are computed by evaluation scripts which store their results in plain GNUPLOT data files. The framework also provides rudimentary support for generating plots from these data that are suited for a rough inspection of the results. More sophisticated plots, e.g., comparing the results of several experiments, can be easily constructed using plotting packages such as GNUPLOT or PGFPLOTS.

5.8. Usage Examples

This section illustrates the intended use of the evaluation framework on some exemplary experiments. A complete implementation of visual SLAM is evaluated on a usual test scenario.

First, the framework is used to generate two test sequences. Both sequences use the same scene, camera, and trajectory but different motion blur settings.

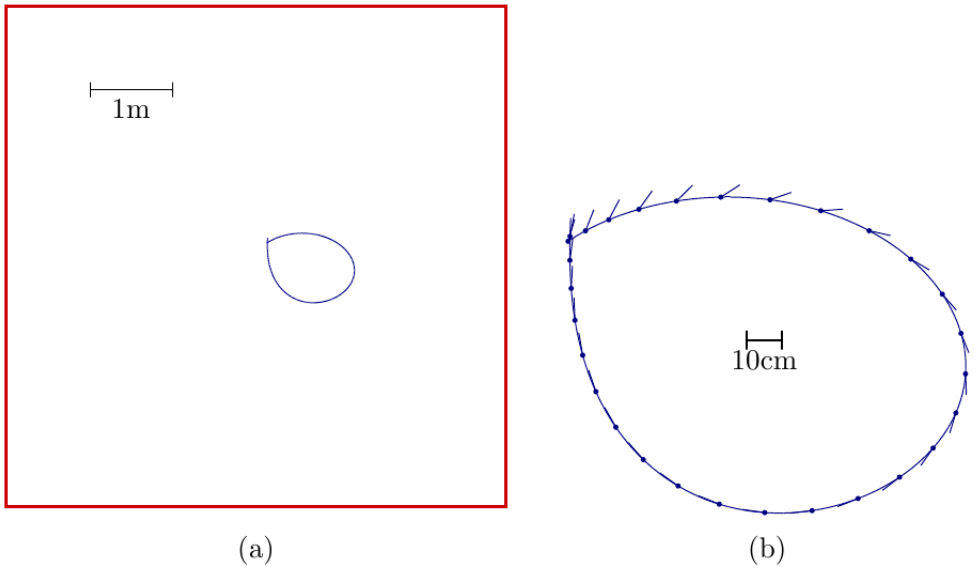


Figure 5.3.: Schematic view of the cube sequence. (a) shows a top view of the scene and trajectory. (b) shows a close-up view of the trajectory. For every 10th frame, the viewing direction is indicated by a vector in the direction of the camera's optical axis.

The scene consists of the inside of a cube of dimension $6\,\mathrm{m} \times 6\,\mathrm{m} \times 6\,\mathrm{m}$. The inside walls of the cube are texture-mapped with photographs of boxes and clutter on a shelf. We use an approximately circular loop trajectory with the camera facing roughly in the direction of motion. The trajectory was obtained by recording a real image sequence using a hand-held stereo camera, reconstructing the camera path, and smoothing the result. Figure 5.3 shows a schematic of the cube scene and trajectory. The camera is modeled after a Point Grey Bumblebee® stereo camera with 640×480 resolution and 65° horizontal field of view.

This setup is used with two different settings for motion blur. The first variant uses no motion blur at all. For the second variant the camera shutter time is set to 0.033 seconds. Because the camera motion is fast, this results in quite severe blurring artifacts, cf. Figure 5.4. Motion blur is generated using 10 camera pose samples per rendered image.

To collect experimental data the visual SLAM implementation described in Chapter 4 is augmented with the C++ API of the framework.[3]

In the first experiment, we simply run the visual SLAM system on both sequences (with and without motion blur). Results of this experiment are shown in Figures 5.5 and 5.6. The plots illustrate some of the information that is produced by the *evaluation system*, namely the evolution of map size, the map error, and camera trajectory errors. The corrupting influence of motion blur on the system performance is clearly visible. Map error as well as absolute camera position and rotation error increase. Especially for the

[3] Another modification that has been made is the use of the inverse depth bundle feature parameterisation instead of the Euclidean parameterisation. This parameterisation will be discussed in Chapter 6.

without motion blur with motion blur

Figure 5.4.: Rendered images from the two versions of the example sequence. On the left, a image rendered without motion blur is shown. On the right, the corresponding image from the sequence with motion blur is depicted. In both cases the right image of the stereo pair is shown.

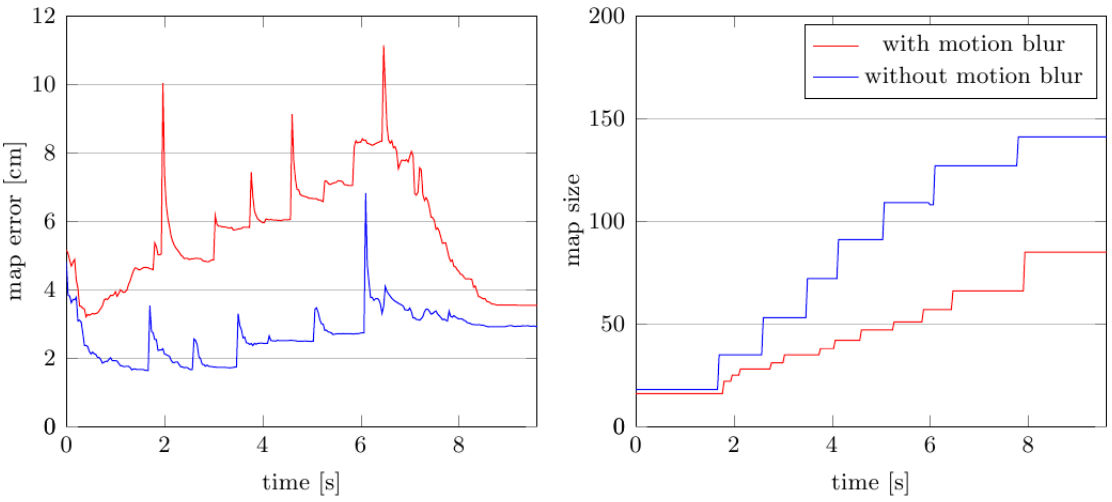


Figure 5.5.: Comparison of map error and state size. The plot on the left compares the map error on the sequences with and without motion blur. The plot on the right compares the evolution of map size over time.

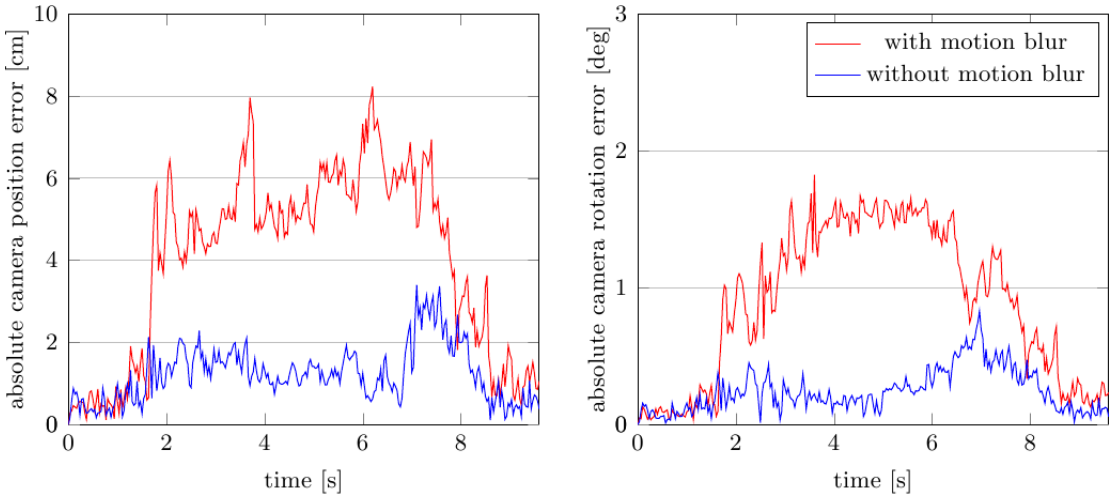


Figure 5.6.: Comparison of trajectory errors. The plot on the left shows the absolute camera position error, the plot on the right shows the absolute camera rotation error. In both plots, the results for the sequence with and without motion blur are compared.

motion-blurred sequence, it is interesting to observe the huge reduction in both map and trajectory error when the loop is closed towards the end of the sequence. Figure 5.5 also shows the evolution of the size of the map over time. The map for the motion-blurred sequence contains fewer features. This is because the FAST detector (Rosten & Drummond, 2006) that we use to select new feature candidates mainly responds to sharp corners, which are smudged by the motion blur.

An advantage of our proposed framework is that it provides access to the *ideal* measurements, i.e., the true projections of feature centers in all images. This allows to analyse the distribution of overall or per-feature measurement errors.

The distribution of measurement error over all features and all frames is compared for the blurred and non-blurred sequence in Figures 5.7(a) and (b), respectively. Each blue dot corresponds to the error of an individual measurement. The ellipses overlaid on the scatter plots correspond to $1, 2$, and 3 standard deviations of the error distributions. As we would expect, the measurement errors for the blurred sequence are larger. The distribution is elongated in the X direction, which is the main blur direction induced by the camera trajectory. Interestingly, in Figure 5.7(b) the measurement error is mainly distributed in a square-shaped region of approximately 1×1 pixel. This can be attributed to the fact that measurements are only obtained with pixel accuracy.

For the next experiment, we modify the measurement method to give subpixel-accurate feature matches. We employ the subpixel refinement procedure described in Section 4.7.1. For the subpixel experiment, we use the non-blurred sequence. The exact same features were used as in the previous experiment. This is achieved by feeding the log of initial 2D feature positions back into the system. The improvement of measurement accuracy achieved by subpixel matching is illustrated in Figure 5.7(c).

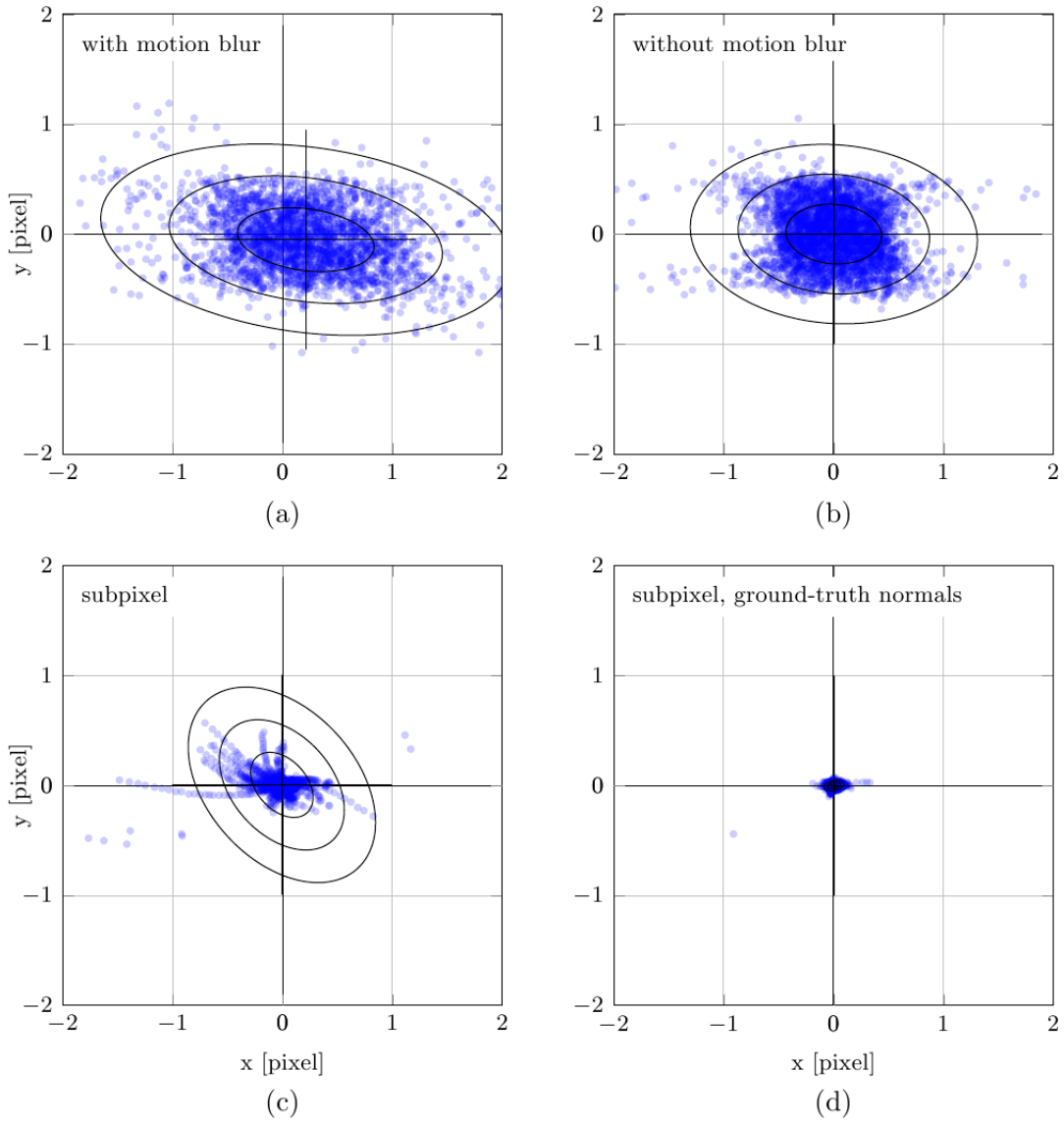


Figure 5.7.: Scatter plots of measurement error. The plots show the distribution of measurement errors in the image plane of the right camera. The motion blurred sequence was used for (a), all other plots were created using the non-blurred sequence. (a) and (b) use the pixel accurate measurement method, whereas (c) and (d) use subpixel-accurate matching. Additionally, for (d) perfectly known feature normals were assumed.

Feature measurement error is in part also caused by incorrectly predicted matching templates due to errors in the assumed normal vectors of feature patches. This effect can be seen in Figure 5.7(c): There are "trails" of measurement error, which correspond to systematic errors in the matching templates.

The final experiment is an example of quick hypothesis checking. We investigate how accurate normal estimation, e.g., using the method proposed by Molton et al. (2004a), would influence measurement accuracy. To achieve this, we add an *evaluation script* which traces the ground truth feature normals. This information is fed back into the visual SLAM system together with the initial 2D feature positions. Figure 5.7(d) illustrates the combined benefit of subpixel accurate measurements and known feature normals. Again, we have used the same features as in the previous experiment. Comparing Figures 5.7(b) and (d) we can conclude that an enormous reduction of measurement error can be expected by the combination of subpixel-accurate matching and exact normal estimation.

5.9. Discussion

We presented an approach to visual SLAM evaluation that falls within the category of simulation, with the obvious benefit of known ground truth. In contrast to point-cloud simulation, there are two advantages. Firstly, the SLAM system is not restricted in its choice of map features because ground truth generation adapts to the systems decisions in this respect. This means that the effects of initialisation and map management heuristics are not ignored. Secondly, the SLAM system operates on images instead of abstract 2D measurements. As we have shown in the previous section, this allows to analyse the effects of phenomena of the imaging process (such as motion blur) and compare implementation choices on the image processing level. Moreover, only minimal changes to the evaluated visual SLAM implementation are required. The code that captures camera images is simply replaced by the interface code provided with the framework.

The capability of analysing individual parts of a visual SLAM system within the SLAMDUNK framework was illustrated. In our experience, the framework is a tremendous help in analysing the influence of phenomena such as motion blur in isolation. Still, careful design of experiments is crucial and in no way trivial. Further experiments can be found throughout the remainder of this thesis, where the framework has enabled thorough experimental validation of our results.

A long-term goal of this work is to create standardised benchmarks upon which different systems can be compared. Ideally, the evaluation should boil down to a single measure of *map quality*. To achieve this, one important question is how to incorporate estimates of map uncertainty. This is a difficult problem, especially because maps differ in number and types of features.

Besides tackling this issue, there are a lot of technical improvements possible. On the one hand, this involves adding more realism to the sequence generation, e.g., better image degradation models like vignetting, shading, Bayer-filter simulation, radial distortion, auto-focus simulation, etc. On the other hand, ground truth generation capabilities should be extended, e.g., for other feature types like planar or line features, and for dynamic scenes.

6

Efficient Point Feature Representation – Inverse Depth Bundles

6.1. Introduction

In this chapter, we propose an efficient representation for point features: inverse depth bundles. By efficient we mean here that with this parameterisation the number of dimensions that are occupied per feature in the state vector is smaller than for other feature parameterisations that have been proposed in the literature. This is an important advantage. The EKF update scales quadratically with the size of the state vector. The bundle parameterisation allows to handle fully correlated maps with significantly more features than was previously possible in real-time. This is achieved by collecting features that were initialised from the same camera image into feature bundles that share large parts of their state representation. For constructing maps with a high point-feature density this is immediately useful because adding more features to a given reference image is inexpensive.

Approaches that handle large maps by breaking them into smaller parts can also benefit from efficient parameterisation. In update amortisation schemes, i.e., postponement (Knight et al., 2001) respectively the Compressed EKF (Guivant & Nebot, 2001), the size of the local working set can be enlarged and the cost of a global update is reduced. In submapping approaches that maintain fully correlated local submaps (e.g., Estrada et al., 2005; Williams et al., 2002; Tardós et al., 2002; Bailey, 2002; Eade & Drummond, 2007) the size of the local maps can be increased or the local update cost can be decreased, respectively.

When we started the work in this chapter, efficient parameterisation was not the primary objective. The main goal was to make accurate predictions of the appearance of point features when seen from new camera poses. To achieve this we have to make assumptions about the local scene structure around the point feature. Here, we make the (quite common) assumption that the point feature lies on a locally planar scene surface.

The warp between two camera images of this surface is a homography transformation (see, for example, Hartley & Zisserman, 2000) which can be parameterised by the orientation of the plane and the poses of the two cameras. A consequence is that for accurate prediction we need to maintain an estimate of the reference camera pose.[1]

In this chapter, we will first introduce the *view-point based* feature parameterisation, which makes this reference camera pose a part of the feature state. The *inverse depth bundle* parameterisation which is discussed subsequently is a simple modification of the view-point based parameterisation which reduces the per-feature state size by sharing the reference pose. The result is an inverse depth parameterisation where a bundle of features shares a common 6 parameter anchor. Only one additional state entry per feature is required, namely the inverse depth of the feature along a fixed ray from the reference pose. The cost of representing the full 6 parameter reference pose is quickly amortised. If 4 or more features are initialised from the same camera frame the representation is smaller than the corresponding Euclidean parameterisation.

The Euclidean feature representation and related models allow three degrees of freedom for every point feature. In the bundle representation, two of these are moved into the shared anchor representation, leaving only one degree of freedom that is maintained independently per feature. An important question is whether this is a sufficient representation or whether the additional degrees of freedom are strictly required. To justify our modeling choices, we present a detailed analysis of the feature measurement process. In particular, we discuss biases introduced by the measurement process.

This chapter is structured as follows. We review related work in Section 6.2. In Section 6.3 we examine point features and how they are measured. We review feature representations using inverse depth in Section 6.4. The view-point based parameterisation, which can be viewed as an extension of the unified inverse depth parameterisation (Montiel et al., 2006; Civera et al., 2008), is introduced in Section 6.5. The inverse depth bundle parameterisation is introduced in Section 6.6. Afterwards, we discuss how to predict appearance for inverse depth bundle features from novel camera views in Section 6.7. In Section 6.8 we address the issue of measurement bias. Section 6.9 presents an experimental evaluation of our feature model on artificial as well as real image sequences. We conclude with a discussion in Section 6.10.

6.2. Related Work

Early visual SLAM respective Structure from Motion systems (Davison, 2003; Chiuso et al., 2002) used the Euclidean feature parameterisation.[2] From the beginning, it was well understood that the Euclidean parameterisation is not well suited to the low-parallax situations occurring with distant or newly initialised features whose depth estimate has not yet converged. Lately, parameterisations using *inverse depth* have been predominantly used. Although similar concepts have been known for a long time in the tracking literature the application of inverse depth parameterisations to SLAM is relatively recent.

[1] The reference camera pose is the camera pose of the image from which the feature template is taken. Usually, that is the camera pose from which the feature was first observed and initialised.

[2] The Euclidean parameterisation was discussed in Section 4.6.

Inverse depth representations were introduced in parallel by Eade & Drummond (2006b) and Montiel et al. (2006).

Eade & Drummond (2006b), in their FastSLAM (Montemerlo & Thrun, 2003) based system use an inverse depth feature representation during an initialisation phase after which features are converted to Euclidean parameterisation. In later work, Eade & Drummond (2007) present a visual SLAM system which builds local submaps that are organised in a graph structure which is globally optimised. A 3-parameter inverse depth feature representation is used, where features are represented with respect to local submap coordinate systems with an inverse z coordinate.

Montiel et al. (2006) propose the *unified inverse depth* parameterisation which handles nearby and distant features in a common framework. Crucially, the approach allows for *undelayed initialisation*: Newly observed features can be immediately inserted in the state and used to improve camera estimates.[3] We review the unified inverse depth parameterisation in Section 6.4. The parameterisations that we propose in this chapter can be viewed as extensions of this work.

One drawback of the unified inverse depth parameterisation is that each feature is represented by 6 parameters, i.e., the feature state vector is twice as large as for the Euclidean representation. Given the quadratic complexity of the EKF with respect to state size, this severely restricts the map size that can be handled in real-time. Paz (2008) addresses the problem in a visual SLAM system using a stereo camera. The system employs inverse depth parameterisation for distant features. To reduce the state size, the Euclidean parameterisation is used for nearby features. Civera et al. (2007) propose to switch inverse depth features to the Euclidean parameterisation once their uncertainty region approaches Gaussianity. Civera et al. (2008) present a real-time implementation and report map sizes of 60 - 70 features using the switching scheme compared to maps of 50 features using exclusively inverse depth features.

An approach to further reduce the state size has been presented by Gee et al. (2007). They detect groups of features lying on a common scene plane. These features can then share a representation of the plane, requiring only two additional state entries per feature to describe the location within the plane. Our inverse depth bundle parameterisation is based on a similar idea. Instead of grouping features by co-planarity, we form groups of features which have been initialized from the same camera frame, i.e., from the same point of view.

Pupilli & Calway (2006) employ a similar representation to ours in a particle filtering SLAM framework (but using depth instead of inverse depth). They also point out the potential decrease in state size, although they do not actively exploit this. We propose a simple feature initialisation strategy to facilitate sharing anchors between many features. To keep the state small we try to minimize the number of camera frames used for feature initialisation, and initialise many features in each of these frames.

In work on Structure from Motion, Azarbayejani & Pentland (1995) use a one-parameter representation that is also built on a similar idea. Every feature is parameterised by its

[3] Originally, Davison (2003) used a delayed initialisation scheme where a separate particle filter was employed to update depth estimates of new features. Features were inserted into the main filter only after the depth uncertainty was small enough to be safely represented by a Gaussian in the Euclidean parameterisation.

depth along a ray with respect to a reference frame. Only a single reference frame is used which is fixed as the first frame of the sequence and is not explicitly represented. Features are only initialised in this first frame which restricts tracking to short sequences where these initial features do not disappear from the image. In contrast, we apply the idea to full SLAM with a map of persistent features. We initialise bundles of features from several reference frames with explicitly represented anchors.

Azarbayejani & Pentland (1995) note that a one-parameter parameterisation may introduce measurement bias but argue that even large biases only have moderate effects on accuracy. Later, Chiuso et al. (2002) argue against one-parameter models and present simulation results which indicate that the biases can have catastrophic effects. We will argue later that these effects are not due to measurement biases but due to another property of their system: They initialise new features as the image sequence progresses but they do *not* initialise new reference frames.

We will discuss measurement bias in more detail in Section 6.8, where we argue that the models underlying the simulations used in (Chiuso et al., 2002) and similar work are biased against one-parameter models which makes their results questionable. We present experiments on rendered images which do not abstract away the measurement process and thus allow a fairer comparison of one-parameter models and models including bias parameters.

For our inverse depth bundle model, we propose to predict feature appearance from novel view points using homography warping. Appearance prediction is used in most current visual SLAM systems. The quality ranges from no warping at all (e.g., Davison, 2003) via affine approximation (e.g., Klein & Murray, 2007) to full homography warping (e.g. Davison et al., 2007). For accurate prediction of the homography, we need accurate estimates of the normal vector of the feature and the reference camera pose. In contrast to other feature models, the reference camera pose is part of the feature state in our inverse depth bundle model.

6.3. Point Feature Appearance and Measurement

In the state estimation machinery of a visual SLAM system, features are commonly abstracted as 3D points. The process of how measurements are obtained is often treated as a black box which, by means of some image processing and data association magic, provides abstract 2D measurements of the image projection of these 3D points.

In this section want to take a closer look at the scene structures corresponding to point features, how point features are measured, and what the underlying assumptions are. It is important to bring these details to mind to motivate the view-point based feature representation which follows in Section 6.5.

6.3.1. Measuring Point Features

Conceptually, a point feature is just that: a 3D point. Conceptually, a measurement of a point feature is a 2D point, the image coordinates of the projection of the 3D point. It is obvious however that such ideal points cannot actually, directly be measured in the camera image because a point has no spatial extent.

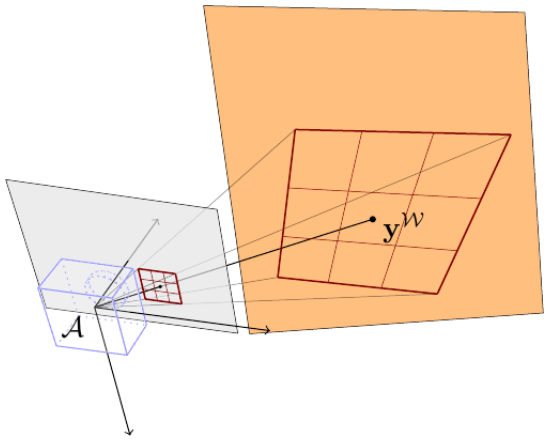


Figure 6.1.: We assume that feature points $\mathbf{y}^{\mathcal{W}}$ lie on locally planar surface patches in the world. When the feature is first observed from camera pose $\mathcal{A}$ a reference template is initialised from the camera image, as indicated by the red grid on the image plane. The actual size of the scene region corresponding to the reference template depends on the distance to $\mathbf{y}^{\mathcal{W}}$.

Commonly, measurements are made by *correlation search*, respectively *template matching*. That is, the image is searched for a matching template of the feature. The matching template is a small image patch (e.g., 11×11 pixels) of how we expect the local neighbourhood of the feature point to appear in the camera image. The image is searched for the best match to this template, i.e., the most similar pixel patch within the search region. The position of the best match provides the 2D measurement of the feature. This procedure has been described in detail in Section 4.7.

Template matching requires that we are able to state how we expect the feature to appear in a given image. This, in turn, requires to make assumptions as to whether and how the appearance of the feature changes with camera pose.

The simplest assumption is that the appearance does not change at all: A reference template patch for the feature is taken from the image when the feature is initialised, and this patch is used to match the feature in all subsequent images. However, this clearly leads to rather inaccurate appearance templates. At least, we would like to be able to model effects such as the following: "When the camera rotates around the Z axis, the patch in the image appears rotated in the opposite direction." (We can predict that this will always occur, even without making any assumptions about the scene.) Or: "When the camera moves away from a feature it will appear smaller."

6.3.2. Point Features as Locally Planar Scene Surfaces

Here, we will make the following assumption about the scene: Point features lie on Lambertian surfaces[4] that can be locally approximated as planar. Here, "locally" refers to the

[4] A surface is *Lambertian* (has Lambertian reflectance) if light falling on it is scattered such that the apparent brightness of the surface is independent on the angle of view. A Lambertian surface will look

region covered by the reference template back-projected to the scene. We further assume constant illumination for the prediction of new matching templates (projections of the reference template into other camera images).[5]

As illustrated in Figure 6.1, the *reference template* is a small pixel patch taken from the reference image, that is, the camera image in which the feature is first observed and initialised. We will refer to the camera pose of this reference image as the *reference pose* $\mathcal{A}$. The reference template has a fixed size in the image, which means that the shape and size of the corresponding scene region depends on the position and orientation of the scene surface. For instance, reference templates for patches initialised on more distant surfaces span larger scene areas.

The image appearance of the feature, respective the reference template, depends on the following:

- the position and orientation of the camera (the reference pose),
- the position and orientation of the plane, and
- the texture of the plane.

More importantly, these parameters are also required to reconstruct the scene appearance from the reference template. We can think of the camera in Figure 6.1 as a projector that re-projects the reference template onto the scene surface.

6.3.3. Feature Appearance from New Camera View-Points

All images of a feature are "generated" by projecting the planar scene surface onto the image plane of a camera. The surface texture is not known exactly. Instead, we have an observation of the texture, namely the reference template. How the reference template maps to the surface depends on several parameters, as discussed above. If we update our estimate of the reference pose $\mathcal{A}$ from which the feature was observed, then this changes our estimate of the surface texture.

The appearance of the feature from the new camera pose $\mathcal{C}$ can be predicted by projecting the reference template onto the scene surface and then projecting from the surface to the image plane of camera $\mathcal{C}$. This is illustrated in Figure 6.2. The re-projection onto the scene surface depends on the position and orientation of the surface as well as on the camera pose $\mathcal{A}$. The projection onto the new image depends on the position and orientation of the surface as well as on the camera pose $\mathcal{C}$.

We denote this transformation between the images as the *warp function*

$$w^{\mathcal{CA}} : \mathbb{R}^2 \to \mathbb{R}^2. \tag{6.1}$$

equally bright from any direction, whatever the direction along which it is illuminated. Examples of such surfaces include cotton cloth, many carpets, matte paper and matte paints (Forsyth & Ponce, 2002, p. 65).

[5] This assumption is mitigated by the use of the *zero-mean* sum of squared differences (ZSSD) criterion in the correlation search. Because the ZSSD is invariant to additive intensity changes, it provides some robustness with respect to changing illumination.

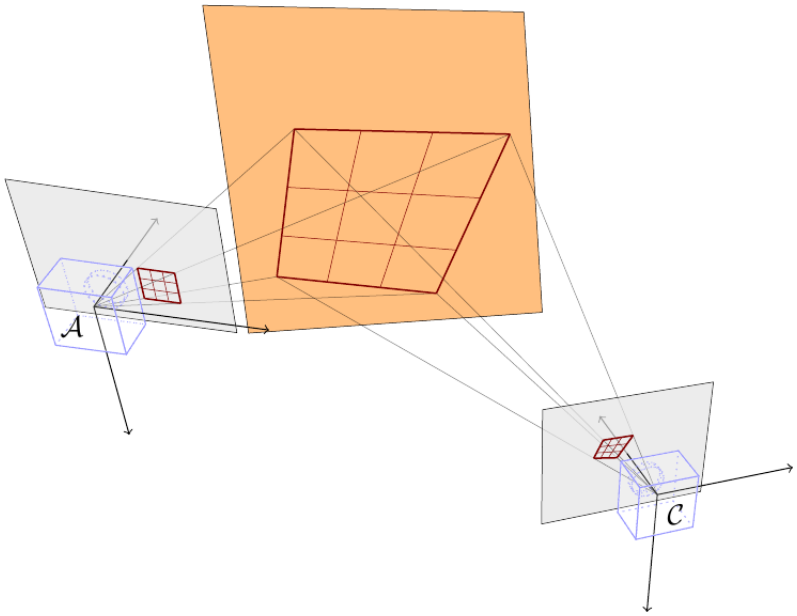


Figure 6.2.: We assume that all views of a feature are projections of a locally planar scene surface. The appearance of the plane surface itself is never directly observed. Rather, it is only observed via its projection in a reference image taken from pose $\mathcal{A}$. The appearance in the image taken from a new pose $\mathcal{C}$ can be obtained by projecting the reference template to the planar surface and further projecting from the planar surface to the image plane of $\mathcal{C}$.

The function maps pixel coordinates $\mathbf{u}^{\mathcal{A}}$ in the reference image to pixel coordinates $\mathbf{u}^{\mathcal{C}} = w^{\mathcal{CA}}\left(\mathbf{u}^{\mathcal{A}}\right)$ in the current image. Under the planar surface assumption, $w^{\mathcal{CA}}$ is a homography transformation induced by the feature plane (Hartley & Zisserman, 2000). This homography can be parameterised by the poses $\mathcal{A}$ and $\mathcal{C}$, and the pose of the plane. We will derive its specific form for our inverse depth bundle parameterisation in Section 6.7.

6.3.4. Feature Appearance Prediction

For correlation search, we are interested in predicting the appearance of a feature from the current camera pose $\mathcal{C}$. We want to create a matching template which we can then search for in the current image to obtain a 2D measurement of the feature. To make an accurate prediction, we need the poses of both, the reference $\mathcal{A}$, and the current camera $\mathcal{C}$. We also need the position and orientation of the plane, where the position is given by any point on the plane, e.g., the features world coordinates $\mathbf{y}^{\mathcal{W}}$. The orientation can be given as a normal vector, for example.

To obtain accurate matching templates, the best estimates of all these parameters should be used. However, most current SLAM systems only keep up-to-date estimates of $\mathcal{C}$ and $\mathbf{y}^{\mathcal{W}}$. In this respect, the models we will discuss represent an improvement as they explicitly contain the reference pose $\mathcal{A}$ as a part of the state vector.

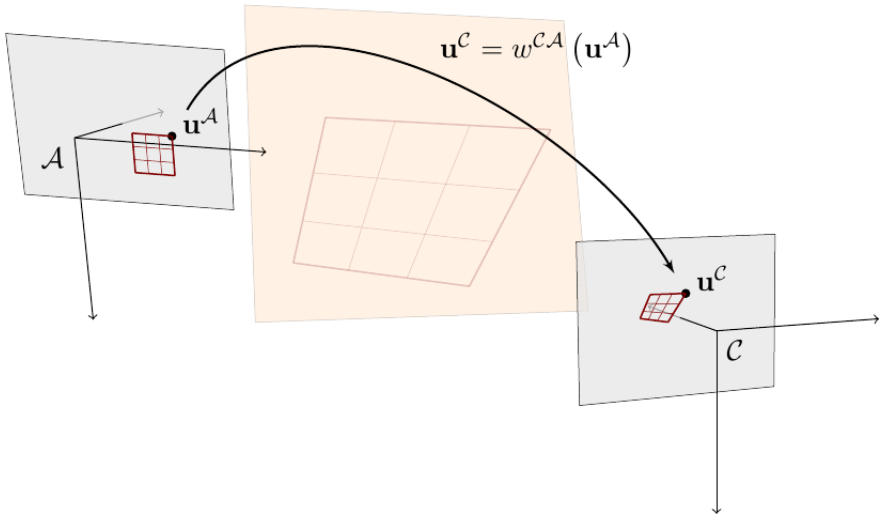


Figure 6.3.: The projection of the reference template to the current image via a planar scene surface can be expressed as a homography transformation $w^{\mathcal{CA}}$ between the image planes.

For the very best results, the following issues should be considered as well:

- Use a more realistic illumination model.
- Estimate the surface orientation, i.e., the normal vector of the feature. An approach to orientation estimation is presented by Molton et al. (2004a), for example.
- Estimate pixel noise in the reference template. The pixel values in camera images are influenced by noise. This pixel noise in the reference template causes a bias in the predicted matching templates. The ideal, noise-free pixel values could be estimated and refined over time.
- Increase the resolution of the reference template image. This, too, can be accomplished within an EKF framework as shown for example by Dellaert et al. (1998).

None of these points are currently addressed in the models presented in this chapter. In particular, the plane normal is not contained in the state vector and hence not updated. The main difficulty lies in that to recover the above data, we would need to go beyond the simple 2D correlation measurements. We will show how this can be accomplished in Chapter 7, where a planar feature model will be introduced which explicitly represents and updates the feature normal.

6.4. Inverse Depth Parameterisations

Two open problems in Davison's original monocular system (Davison, 2003) were undelayed feature initialisation and handling of distant features. These issues are connected and both result from the shape of the probability density of a feature's 3D position when

the feature has only been observed under small parallax angles.[6] These low-parallax situations occur for features that are very distant in relation to the translational range of the camera motion. For example, there is no measurable parallax in observations of a star even if the camera translates for many kilometers. Moreover, low-parallax situations occur for newly initialised features where during the first few observations the camera translation is small relative to the depth even for nearby features.

The probability density of the 3D position of such features is not approximated well by a Gaussian distribution. Civera et al. (2008, p. 933) note that "the depth coordinate of such features has a probability density that rises sharply at a well-defined minimum depth to a peak, but then, tails off very slowly toward infinity – from low parallax measurements, it is very difficult to tell whether a feature has a depth of 10 units rather than 100, 1000 or more." Furthermore, the measurement model is highly nonlinear in the depth (respective Z) coordinate of the feature due to the perspective projection. This is especially harmful during the first few observations of a feature when uncertainty is still very high.

Due to this phenomenon, undelayed initialisation of new features into the map is unfeasible when the feature is parameterised in Euclidean coordinates (Solà et al., 2005; Eade & Drummond, 2006b; Montiel et al., 2006; Civera et al., 2008). In monocular SLAM in particular, the first observation provides no information at all about the depth of a new feature. The probability density of the new feature's 3D position is cone-shaped with the apex on the camera center and the axis going through the first observation to infinity. Observations with sufficient parallax are necessary to constrain the depth until the density becomes sharply peaked and can be approximated as Gaussian.

In a stereo setting, initialisation is unproblematic for nearby features. From a stereo camera we have a disparity measurement for a new feature. For features that are relatively close this allows for undelayed initialisation, even for the Euclidean parameterisation.[7] For distant features however this advantage is lost: The disparity provides little information because the baseline is small relative to the feature depth.

According to the stereo projection function (3.30) the disparity d of a point with depth z is

$$d = f_x \frac{b}{z}. \tag{6.2}$$

The dependence between the disparity and the depth is nonlinear. When initialising a new feature the measurement uncertainty in the disparity value is propagated through this nonlinear function to give an estimated depth uncertainty. While it is reasonable to assume Gaussian distributed disparity error, the corresponding distribution when projected through the nonlinearity is not Gaussian, especially for features with small disparity respectively large depth. However, the above equation is linear in the *inverse depth* $\rho = z^{-1}$. The Gaussian disparity uncertainty corresponds and propagates to Gaussian uncertainty in inverse depth.

[6] The parallax between two observations of a point is the angle between the viewing rays from the respective camera center to the point.

[7] The meaning of "relatively close" depends on the baseline of the stereo camera. Paz (2008) uses the same stereo camera in their work as we use here. She experimentally determines the threshold for Euclidean parameterisation as $\leq$ 5m.

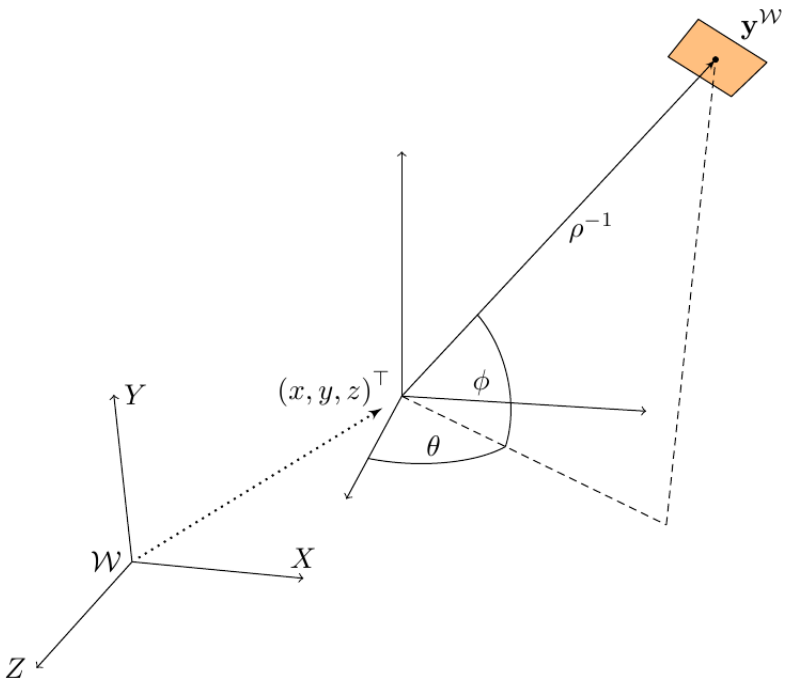


Figure 6.4.: The unified inverse depth parameterisation. The world coordinates of the feature $\mathbf{y}^{\mathcal{W}}$ are at inverse depth ρ along a ray from $(x, y, z)^\top$ in the direction encoded by polar angles θ, ϕ.

Inverse depth feature parameterisations have been independently proposed by Eade & Drummond (2006b) and Montiel et al. (2006). Both authors offer the same arguments about the advantages of inverse depth representation: The measurement equation is nearly linear in the inverse depth, as long as the camera rotation is mostly around the optical axis and the displacement along the optical axis is small relative to the depth. This allows for undelayed feature initialisation in monocular SLAM. [8]

Montiel et al. (2006) propose the *unified inverse depth* parameterisation which we will briefly review here because the parameterisations we propose subsequentially can be viewed as an extension of this approach.

In the unified inverse depth parameterisation a feature is represented by a 6-vector $\mathbf{y} = (x, y, z, \theta, \phi, \rho)^\top$. Here, the point $(x, y, z)^\top$ is the camera position from which the feature was initialised. The polar angles θ and ϕ encode a ray from this point in the direction of the feature. Finally, ρ is the inverse depth of the feature along this ray. The

[8] For undelayed initialisation, the initial inverse depth mean and variance are set such that the 95% confidence region spans a depth range from close to the camera up to infinity. Civera et al. (2008, p. 938) note that "experimental validation has shown that the precise values of these parameters are relatively unimportant to the accurate operation of the filter as long as infinity is clearly included in the confidence interval."

parameters are illustrated in Figure 6.4. The Euclidean point encoded by $\mathbf{y}$ is

$$\mathbf{y}^{\mathcal{W}} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \frac{1}{\rho} \mathbf{m}(\theta, \phi) \tag{6.3}$$

where $\mathbf{m}(\theta, \phi)$ denotes the unit vector in the direction given by θ, ϕ.

It is worth noting that points at infinity, i.e., with $\rho = 0$, can be handled naturally. It might seem at first, that the $\frac{1}{\rho}$ factor presents a problem. However, this apparent difficulty disappears when the point is projected into the image. For the projection of a point in a monocular camera, only the direction to the point is important, not the scale. So in this case, the scale factor can be dropped.

Stereo Projection for Inverse Depth Points For projection in a stereo camera we can not simply disregard the scale factor because the (inverse) depth determines the disparity. Consider a 3D point $\mathbf{x} = \frac{1}{\rho}(x,\, y,\, z)^\top$ given by a directional vector (not necessarily of unit length) and an inverse scale factor ρ. The stereo projection (Equation 3.30) of this point is

$$\pi_s(\mathbf{x}) = \pi_s \begin{pmatrix} \rho^{-1}x \\ \rho^{-1}y \\ \rho^{-1}z \end{pmatrix} = \begin{pmatrix} f_x \frac{\rho^{-1}x}{\rho^{-1}z} + u_0 \\ f_y \frac{\rho^{-1}y}{\rho^{-1}z} + v_0 \\ f_x \frac{b}{\rho^{-1}z} \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \\ \rho f_x \frac{b}{z} \end{pmatrix} \tag{6.4}$$

We see that the scale factor cancels out in the first two rows, i.e., the "monocular" part of the projection. The scale factor appears inverted in the last row. As expected, points at infinity (with $\rho = 0$) have zero disparity.

Let us define a modified stereo projection function which has both the inverse depth and the (unscaled) point coordinates as arguments.

$$\pi_\rho : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3 : \left(\rho,\, \begin{pmatrix} x \\ y \\ z \end{pmatrix} \right) \mapsto \begin{pmatrix} u \\ v \\ d \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \\ \rho f_x \frac{b}{z} \end{pmatrix} \tag{6.5}$$

Note, that points at infinity with $\rho = 0$ are handled naturally.

6.5. View-Point Based Inverse Depth Parameterisation

Based on the idea of the unified inverse depth parameterisation (Montiel et al., 2006), we introduce a new feature representation. We will refer to this parameterisation as *view-point based* because it describes features in terms of the initial view-point, i.e., the camera pose at the time of initialisation.

A view-point based feature consists of a feature state $\mathbf{y}$, a feature ray $\mathbf{m}^{\mathcal{A}}$, and a template image T. In the probabilistic state, the feature is represented by the 7-dimensional vector

$$\mathbf{y} = \begin{pmatrix} \mathbf{c} \\ \boldsymbol{\phi} \\ \rho \end{pmatrix}. \tag{6.6}$$

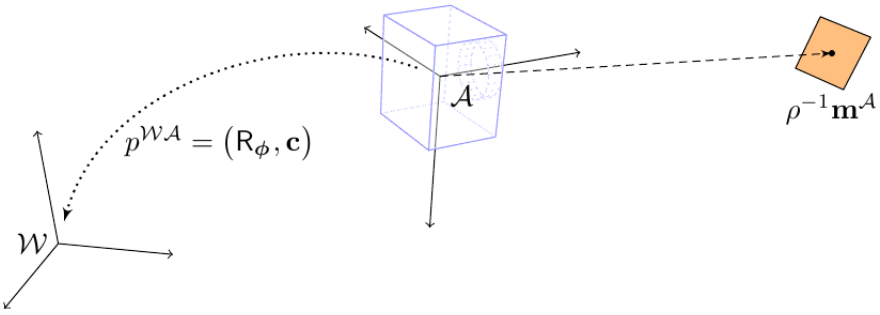


Figure 6.5.: View-point based feature model. The relative pose of world frame $\mathcal{W}$ and anchor (initialisation camera) frame $\mathcal{A}$ is given by translation $\mathbf{c}$ and rotation $\boldsymbol{\phi}$. The unit vector $\mathbf{m}^{\mathcal{A}}$ defines a ray to the feature centre, and ρ is the inverse depth along this ray. Importantly, the ray $\mathbf{m}^{\mathcal{A}}$ is represented relative to the anchor frame. It is accurately known despite the uncertain pose of the initialisation camera frame.

Here, the 3-vector $\mathbf{c} = (x,\, y,\, z)^\top$ is the camera position at the time of the first observation of the feature. The 3-vector $\boldsymbol{\phi} = (\phi_x,\, \phi_y,\, \phi_z)^\top$ is an exponential rotation representing the camera rotation for this first observation. Together, $\boldsymbol{\phi}$ and $\mathbf{c}$ define the pose of the *anchor* (the initialisation camera coordinate frame) with respect to the world reference system, $p^{\mathcal{WA}} = (\mathsf{R}_\phi, \mathbf{c})$. Finally, ρ is the inverse depth of the feature on a ray in direction $\mathbf{m}^{\mathcal{A}}$.

This ray to the feature is represented in the anchor coordinate frame $\mathcal{A}$. Thus, the unit vector $\mathbf{m}^{\mathcal{A}}$ simply encodes the direction through the pixel where the feature was detected in the initial image. With respect to the reference camera pose, there is no uncertainty as to where the projection of the feature was observed. Thus, $\mathbf{m}^{\mathcal{A}}$ is a (per feature) fixed component of the model and not part of the probabilistic state vector. Furthermore, a template T of the appearance of the feature in the reference image is stored. The view-point based feature model is illustrated in Figure 6.5.

Future measurements of the feature are obtained by performing correlation search within gated search regions in the camera images. Prior to correlation search, the template T is warped to account for varying appearance caused by view-point changes. For this purpose, we assume that T results from the projection of a locally planar scene surface with known normal vector. An estimate of the normal could be obtained by analysing stereo disparities in the neighbourhood of the feature during initialisation. The normal estimate might also be sequentially updated in a separate filter (Molton et al., 2004a). However in our current implementation, we simply assume that the scene surface is facing the camera, i.e., the normal vector is $-\mathbf{m}^{\mathcal{A}}$.

There are two key differences between our view-point based model and the unified inverse depth parameterisation proposed by Montiel et al. (2006). First, by parametrising the full camera rotation $\boldsymbol{\phi}$ one additional degree of freedom is introduced, namely rotation about the ray to the feature. This is not observable from point measurements (u, v, d) of the feature directly. However, via its perfect correlation to the camera rotation estimate at the time of initialisation it becomes correlated to other state variables. Hence, additional

information on rotation about the ray to feature $\mathbf{y}$ is provided by measurements of *other* features. For future measurements, the template T is warped to account for varying appearance caused by view-point changes. Because the full rotation $\boldsymbol{\phi}$ is used in warping the template, updating its estimate can improve the accuracy of the predicted feature appearance for correlation search.

The second difference occurs with respect to initializing uncertainty of the feature ray. In (Montiel et al., 2006), this results from a combination of uncertainty in the initial camera position and measurement uncertainty in the initial $(u, v)^\top$ observation. This rests on the assumption that the initial measurement is subject to the same measurement error as any other measurement. This is justifiable if measurements are of some directly observable physical quantity, like laser range finder measurements of the distance to a wall. We argue, that for the case of making image measurements by correlation search the situation is different. The feature template is the projection of a scene surface in the initial camera image, *not* the scene surface itself. The measurement process proceeds by back-projecting the feature template to the (uncertain) scene surface and then projecting it to the (uncertain) current camera frame, where the projection is used for correlation search. However, if the current camera pose is the initialisation pose, this will always result exactly in the observed initial template *regardless* of feature depth or scene structure. The location of the template in the initial image is known with absolute certainty. To put it another way: We obtain feature measurements using correlation search. In this sense the initial observation is not a measurement at all. It merely provides the tools to carry out measurements in the future.

Hence, we model the initial uncertainty of the feature ray as resulting from the camera pose uncertainty only. We do not assume uncertainty in the pixel position $(u, v)^\top$ for the initial observation.[9] Thus, relative to the anchor coordinate frame, $\mathbf{m}^{\mathcal{A}}$ is fixed. The uncertainty in the feature ray (with respect to the world coordinate frame) is purely a consequence of the uncertainty in pose of the anchor coordinate frame.

6.5.1. Measurement Model

We restrict the following derivation of the measurement model to the measurement of a single feature $\mathbf{y}_i$. We omit the feature index i to avoid cluttered notation. The extension to the case of multiple features is trivially made by stacking individual measurements into a joint measurement vector. This is completely analogous to the detailed discussion for the Euclidean measurement model in Section 4.6.

Before we discuss the detailed generative model, we will, on an abstract level, walk through the steps that transform a feature from the state representation to the image measurement (the 2D image projection of the feature point). The 3D coordinates of the feature point in the anchor coordinate frame are found at depth ρ^{-1} along the feature ray

$$\mathbf{y}^{\mathcal{A}} = \frac{1}{\rho}\mathbf{m}^{\mathcal{A}}. \tag{6.7}$$

[9] If we would correctly model the initialisation errors for the case of correlation search, we should include the pixel intensities of the template in the state vector and initialise their uncertainty with the variance of the intensity noise introduced by the camera.

These coordinates are then transformed to the world reference frame using the pose transform $p^{\mathcal{WA}}$, which is given by the $\boldsymbol{\phi}$, $\mathbf{c}$ components of the feature state. This gives the feature's world coordinates

$$\mathbf{y}^{\mathcal{W}} = p^{\mathcal{WA}}\left(\mathbf{y}^{\mathcal{A}}\right). \tag{6.8}$$

The world coordinates are then further transformed to the coordinate system of the current camera pose,

$$\mathbf{y}^{\mathcal{C}} = p^{\mathcal{CW}}\left(p^{\mathcal{WA}}\left(\mathbf{y}^{\mathcal{A}}\right)\right) = p^{\mathcal{CW}} \circ p^{\mathcal{WA}}\left(\mathbf{y}^{\mathcal{A}}\right) = p^{\mathcal{CA}}\left(\mathbf{y}^{\mathcal{A}}\right). \tag{6.9}$$

These camera coordinates are projected to the image plane to obtain the coordinates of the feature projection in the current image

$$\mathbf{y}^{\mathcal{I}} = \pi\left(p^{\mathcal{CA}}\left(\mathbf{y}^{\mathcal{A}}\right)\right). \tag{6.10}$$

Finally, this ideal projection is distorted by additive measurement error to give the measurement

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{y}^{\mathcal{I}} + \boldsymbol{\delta}. \tag{6.11}$$

Let's look at the model in more detail now. We start by deriving the pose $p^{\mathcal{CW}} \circ p^{\mathcal{WA}}$ that transforms from anchor to camera coordinates. The first factor is the inverse of the camera pose in the state vector

$$p^{\mathcal{CW}} = p^{\mathcal{WC}\,-1} = \left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}},\ \mathbf{r}^{\mathcal{WC}}\right)^{-1} = \left(\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}},\ -\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\,\mathbf{r}^{\mathcal{WC}}\right). \tag{6.12}$$

while the second is the anchor pose described by $\boldsymbol{\phi}, \mathbf{c}$ in the feature state

$$p^{\mathcal{WA}} = \left(\mathsf{R}_{\boldsymbol{\phi}},\ \mathbf{c}\right). \tag{6.13}$$

Their composition is (see Equation 3.3)

$$\begin{aligned} p^{\mathcal{CA}} &= p^{\mathcal{CW}} \circ p^{\mathcal{WA}} && (6.14)\\ &= \left(\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathsf{R}_{\boldsymbol{\phi}},\ \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathbf{c} + \left(-\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathbf{r}^{\mathcal{WC}}\right)\right) && (6.15)\\ &= \left(\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathsf{R}_{\boldsymbol{\phi}},\ \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)\right). && (6.16) \end{aligned}$$

Next, we apply the resulting transformation to the feature's anchor coordinates, giving

$$\begin{aligned} \mathbf{y}^{\mathcal{C}} &= p^{\mathcal{CA}}\left(\mathbf{y}^{\mathcal{A}}\right) && (6.17)\\ &= \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathsf{R}_{\boldsymbol{\phi}}\frac{1}{\rho}\mathbf{m}^{\mathcal{A}} + \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right) && (6.18)\\ &= \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\frac{1}{\rho}\mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}} + \left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)\right) && (6.19)\\ &= \frac{1}{\rho}\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}} + \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)\right). && (6.20) \end{aligned}$$

In the final line we have pulled the $\frac{1}{\rho}$ factor to the front of the expression. We readily apply the modified stereo projection (6.5) to obtain the projected point

$$\mathbf{y}^{\mathcal{I}} = \pi_{\rho}\left(\rho\,,\ \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}} + \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)\right)\right). \tag{6.21}$$

Finally, measurement noise is added to the projected point. We assume zero-mean Gaussian noise $\boldsymbol{\delta} \sim \mathcal{N}(0, \mathsf{R})$. In practice, we use a diagonal noise covariance matrix

$$\mathsf{R} = \begin{bmatrix} \sigma_u^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_d^2 \end{bmatrix}. \tag{6.22}$$

The variances in u and v direction depend on the accuracy of the template matching procedure. The disparity is the u difference between the matches found in the left and right camera image. Thus, assuming uncorrelated matching errors between left and right image we have $\sigma_d^2 = 2\sigma_u^2$.

In summary, after we add the noise term, we obtain the generative measurement model

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{y}^{\mathcal{I}} + \boldsymbol{\delta} = \pi_\rho \left(\rho \,,\, \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1} \left(\mathsf{R}_{\boldsymbol{\phi}} \mathbf{m}^{\mathcal{A}} + \rho \left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}} \right) \right) \right) + \boldsymbol{\delta}. \tag{6.23}$$

To apply the model in the EKF update it must be linearised. We need the Jacobian matrices $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}}$ evaluated at $\mathbf{x} = \overline{\boldsymbol{\mu}}_t$, $\boldsymbol{\delta} = \mathbf{0}$. We note that the measurement of a feature $\mathbf{y}_i$ only depends on this feature's state and the current camera pose. It is independent of other features in the map. This will result in a sparse Jacobian layout. The Jacobian matrices are derived in Appendix A.4.

6.5.2. Initialisation of New Features

Now we want to discuss how view-point based features are initialised into the probabilistic map. Let us suppose that we want to initialise a new feature in the current frame. As input to the initialisation we have two pieces of information.

First, we have an image coordinate

$$\mathbf{u}^{\mathcal{I}} = \begin{pmatrix} u \\ v \end{pmatrix} \tag{6.24}$$

in the right (reference) image of the stereo pair. How this point is chosen is not important for the moment. For instance, the image coordinates $\mathbf{u}^{\mathcal{I}}$ can correspond to a salient image region supplied by a corner detector.

Second, we have a disparity measurement $\mathbf{z}$. The disparity measurement is obtained by correlation search. A small image patch of the reference (right) image around $\mathbf{u}^{\mathcal{I}}$ is searched for in the second (left) image. The pixel offset at which the highest correlation occurs is the measured disparity. We assume that $\mathbf{z}$ is a noisy observation of the hypothetical perfect measurement d, that is, the actual disparity of the point where the ray through $\mathbf{u}^{\mathcal{I}}$ hits the scene. The measurement is modeled as

$$\mathbf{z} = d + \delta \tag{6.25}$$

where the noise term δ is assumed zero-mean Gaussian with variance σ_d^2. Thus, given the disparity measurement our belief about the true disparity is

$$p(d \,|\, \mathbf{z}) = \mathcal{N}(d\,;\, \mathbf{z}, \sigma_d^2). \tag{6.26}$$

It is important to realise the fundamental difference in the nature of $\mathbf{u}^{\mathcal{I}}$ and $\mathbf{z}$. The former is *not* a measurement. Rather, it is a definition, an agreement: we determine that this is the direction along which we want to put the feature. The latter is a true measurement which tells us something about where this ray we defined hits the scene.

Initialising the new feature involves two tasks. First, we have to fix the "static" parts of the feature description namely the ray $\mathbf{m}^{\mathcal{A}}$ and the template image T. Second, we have to augment the probabilistic map state with a new feature vector $\mathbf{y}$.

For the feature template T we store from the current camera image a local image patch around $\mathbf{u}^{\mathcal{I}}$. The patch is taken from the right (reference) image of the stereo pair.

The ray to the feature $\mathbf{m}^{\mathcal{A}}$ is represented in the anchor coordinate frame, that is, in the coordinate frame attached to the camera. It is determined by back-projecting a unit ray through $\mathbf{u}^{\mathcal{I}}$. Because the image coordinate $\mathbf{u}^{\mathcal{I}}$ is also relative to the camera coordinate system, the result is not dependent on the camera pose. We have

$$\mathbf{m}^{\mathcal{A}} = \pi^{+}\left(\mathbf{u}^{\mathcal{I}}\right), \tag{6.27}$$

with

$$\pi^{+} : \mathbb{R}^2 \to \mathbb{R}^3 : \begin{pmatrix} u \\ v \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{1}{\sqrt{1 + \left(\frac{u-u_0}{f_x}\right)^2 + \left(\frac{v-v_0}{f_y}\right)^2}} \begin{pmatrix} \frac{u-u_0}{f_x} \\ \frac{v-v_0}{f_y} \\ 1 \end{pmatrix}. \tag{6.28}$$

The function π^{+} is a back-projection to a point on the plane $z = 1$ which is then normalised to unit length.

To augment the state vector with a new feature $\mathbf{y}$ (see Section 4.8.2) we need the inverse measurement model

$$\mathbf{y} = \mathbf{g}(d, \mathbf{x}) \tag{6.29}$$

which computes the new feature vector as a function of the (ideal, noise-free) measurement and the current state. The feature vector consists of the initial observation pose $\mathbf{c}$, $\boldsymbol{\phi}$, and the inverse depth ρ. The initial observation pose is a copy of the current camera pose.[10]

$$\mathbf{c} = \mathbf{r}^{\mathcal{WC}} \tag{6.30}$$

$$\boldsymbol{\phi} = \log\left(\mathbf{q}^{\mathcal{WC}}\right) \tag{6.31}$$

The inverse depth ρ can be computed from the disparity measurement. According to the stereo projection function (3.30), the disparity of a point $(x,\, y,\, z)^{\top}$ is

$$d = f_x \frac{b}{z} \tag{6.32}$$

We plug in the feature position $\mathbf{y}^{\mathcal{A}} = \rho^{-1}\mathbf{m}^{\mathcal{A}}$, solve for ρ, and obtain

$$\rho = d\frac{z_m}{f_x b}, \tag{6.33}$$

where z_m denotes the Z-coordinate of $\mathbf{m}^{\mathcal{A}}$.

[10] with the rotation converted from quaternion to exponential representation

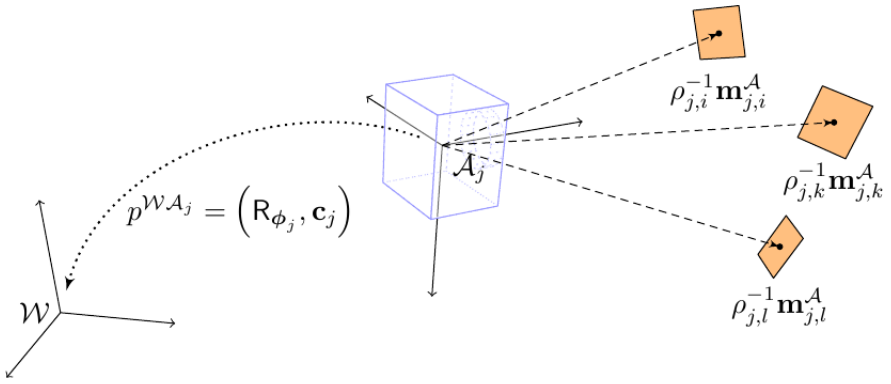


Figure 6.6.: Inverse depth bundle model. The anchor $\mathbf{a}_j = \left(\mathbf{c}_j, \boldsymbol{\phi}_j\right)^\top$ represents the relative orientation of the world frame $\mathcal{W}$ and the anchor frame $\mathcal{A}_j$ by the translation $\mathbf{c}_j$ and the rotation $\boldsymbol{\phi}_j$. Features $\mathbf{y}_{j,i}$ initialised with respect to this anchor are each represented by their inverse depths $\rho_{j,i}$ along rays $\mathbf{m}_{j,i}^{\mathcal{A}}$.

In summary, we have the inverse measurement model

$$\mathbf{y} = \mathbf{g}\left(d, \mathbf{x}\right) = \begin{pmatrix} \mathbf{r}^{\mathcal{W}\mathcal{C}} \\ \log\left(\mathbf{q}^{\mathcal{W}\mathcal{C}}\right) \\ d\frac{z_m}{f_x b} \end{pmatrix}. \tag{6.34}$$

The Jacobians $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{g}}{\partial d}$ are derived in Appendix A.5. Again, the Jacobian by the state vector is sparse, because $\mathbf{g}$ only depends on the components describing the camera state.

6.6. Inverse Depth Bundle Parameterisation

We note that the anchor poses of two view-point based features that are initialised at the same time t, i.e., from the same camera pose, are identical. Both represent the camera pose at time t and thus the respective parts of the state vector are perfectly correlated to each other. It is obvious that it is unnecessary that every feature maintains its own representation of these identical anchors.

We propose the *inverse depth bundle parameterisation* as a straightforward modification of the view-point based model: We group features which are initialised from the same camera image in feature bundles that share a common anchor representation, as illustrated in Figure 6.6. The representation of a feature in the state vector is split into an individual part containing only the inverse depth and an anchor part that is shared among all features of the same bundle.

In the state, an anchor is a 6-dimensional vector

$$\mathbf{a}_j = \begin{pmatrix} \mathbf{c}_j \\ \boldsymbol{\phi}_j \end{pmatrix}, \tag{6.35}$$

where $\mathbf{c}_j$ and $\boldsymbol{\phi}_j$ represent the camera position and rotation of the camera at the time when the feature bundle j was initialised.

The bundle j consists of n_j features $\mathbf{y}_{j,i}$ with $i \in \{1, \ldots, n_j\}$. In the state vector every feature takes only one dimension: the inverse depth

$$\mathbf{y}_{j,i} = \rho_{j,i}. \tag{6.36}$$

The full state vector with m feature bundles then has the form

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_v, & \mathbf{a}_1, & \mathbf{y}_{1,1}, & \ldots & \mathbf{y}_{1,n_1}, & \mathbf{a}_2, & \ldots & \mathbf{a}_m, & \mathbf{y}_{m,1}, & \ldots & \mathbf{y}_{m,n_m} \end{pmatrix}^\top. \tag{6.37}$$

The inverse depth for a feature $\mathbf{y}_{j,i}$ is measured along the feature ray $\mathbf{m}^{\mathcal{A}}_{j,i}$ which is represented with respect to the j-th anchor coordinate frame $\mathcal{A}_j$. We also store a template patch $T_{j,i}$ from the reference image for every feature. Just like for the view-point based representation, the feature rays and templates are fixed components of the model. They are not part of the EKF state vector.

Obviously, the bundle parameterisation will reduce the effective per-feature state size for features that share their anchor. A bundle of n features occupies $6 + n$ entries in the state (6 for the anchor and n for the inverse depths). This means, that for $n \geq 2$ the bundle parameterisation is more efficient than the unified inverse depth parameterisation by Montiel et al. (2006). For $n > 3$ the bundle parameterisation is more efficient than the straightforward Euclidean feature parameterisation. The actual effectiveness of the bundle representation depends on sharing anchors between as many features as possible. In practice this means that the effectiveness depends on the initialisation heuristics because anchors can only be shared between features that are initialised simultaneously.

6.6.1. Measurement Model

The measurement model is exactly the same as for the view-point based model (see Equation 6.23). The only difference is that now $\mathbf{c}$ and $\boldsymbol{\phi}$, and ρ are gathered from the anchor $\mathbf{a}_j$ and feature $\mathbf{y}_{j,i}$ respectively, while before they all were part of the feature $\mathbf{y}$. For the Jacobians see Appendix A.6.

6.6.2. Initialisation of New Anchors and Features

The initialisation of new features is basically the same as for view-point based features. The process is just split into initialisation of anchors and initialisation of features. Nevertheless, it is interesting to look at a few details. When we look at the inverse measurement equation for view-point based features (6.34) the conceptual distinction between the anchor part and the feature part becomes very clear. The anchor is initialised independent of any measurement. Because it is just a copy of the current camera pose it only depends on the current state. On the other hand, the inverse depth (the feature state) is initially completely uncorrelated to the state because it is represented with respect to the current camera coordinate frame.

To augment the state with a new anchor $\mathbf{a}_j$ we write the $\mathbf{a}_j$ as a function of the state vector (see first two rows in Equation 6.34)

$$\mathbf{a}_j = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} \mathbf{c}_j \\ \boldsymbol{\phi}_j \end{pmatrix} = \begin{pmatrix} \mathbf{r}^{\mathcal{WC}} \\ \log\left(\mathbf{q}^{\mathcal{WC}}\right) \end{pmatrix}. \tag{6.38}$$

The new anchor is appended to the state vector and the covariance matrix is updated as

$$\boldsymbol{\mu}_{new} = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{f}(\boldsymbol{\mu}) \end{pmatrix}, \quad \boldsymbol{\Sigma}_{new} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{\Sigma} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top \\ \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \boldsymbol{\Sigma} & \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \boldsymbol{\Sigma} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^\top \end{bmatrix}. \tag{6.39}$$

If an anchor $\mathbf{a}_j$ has been added for the current image, we can initialise features $\mathbf{y}_{j,i}$ as follows. For each feature we have an image coordinate $\mathbf{u}_i^{\mathcal{I}}$ and a noisy disparity measurement $\mathbf{z}_i = d_i + \delta$ at this coordinate.

We initialise the feature ray $\mathbf{m}_{j,i}^{\mathcal{A}}$ according to Equation 6.27 and a template patch T is copied from the image. The feature state $\mathbf{y}_{j,i}$, i.e., the inverse depth, is computed from the disparity as (see last row in Equation 6.34)

$$\mathbf{y}_{j,i} = \rho_{j,i} = \mathbf{g}(d_i) = d_i \frac{z_m}{f_x b} \tag{6.40}$$

where z_m is the z coordinate of $\mathbf{m}_{j,i}^{\mathcal{A}}$. The new feature state does not depend on the state vector. In particular, it does not depend on the new anchor $\mathbf{a}_j$. It is appended to the state vector as

$$\boldsymbol{\mu}_{new} = \begin{pmatrix} \boldsymbol{\mu} \\ \mathbf{g}(\mathbf{z}_i) \end{pmatrix}, \quad \boldsymbol{\Sigma}_{new} = \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \sigma^2_{\mathbf{y}_{j,i}} \end{bmatrix}. \tag{6.41}$$

The covariance of the new feature state is computed from the measurement noise covariance as

$$\sigma^2_{\mathbf{y}_{j,i}} = \frac{\partial \mathbf{g}}{\partial d_i} \sigma_d^2 \frac{\partial \mathbf{g}}{\partial d_i}^\top. \tag{6.42}$$

For the Jacobians see Appendix A.7.

6.6.3. Initialisation Strategy

The inverse depth bundle parameterisation is designed to lower per-feature state size by sharing an anchor between several features. As long as there is a minimum of 3 features per anchor, the bundle representation is at least as efficient as the Euclidean parameterisation. However, of course we should try to do better than that.

The actual gain depends on the strategy employed to decide when to initialise new features. This strategy should be designed to minimize the number of anchors and ensuring that each anchor is shared by many features.

For our experiments we use a simple initialisation heuristic as follows. The camera image area is divided into a 4×4 grid. While making feature measurements in each new image we determine the number of empty grid cells. A grid cell is counted as empty if either, there are no features predicted to be visible in this cell, or, all attempts to measure visible features in this cell failed. If the fraction of empty cells is larger than a threshold (70% in our experiments) a new bundle of features is initialised. New feature candidates are selected to lie on salient image areas and to be evenly distributed in the image. At most 20 new features are initialised per bundle. In Section 6.9 we demonstrate that even with this simple strategy the bundle parameterisation allows to sustain real-time operation for maps of more than 200 features.

The threshold on the number of features per bundle has been chosen empirically to maximise the size of the environment covered by the map. We have experimented with different thresholds on several pre-recorded image sequences. It turns out, that initialising around 20 features per anchor yields the minimal state size for the map. If we use less features this leads to a more frequent initialisation and thus more anchors in the map. At the same time, the overall number of features in the map does not decrease significantly, thus leading to a larger state vector. On the other hand, using more than about 20 features does not further reduce the number of anchors that are required. This leads to a denser map and a larger state vector. If our main goal is to increase map density then we should initialise many features per anchor, of course. However, our main objective here is to demonstrate that anchor sharing decreases the state size even for sparse maps. Thus, we aim to increase the area spanned by the map.

More complex strategies can be envisioned to increase efficiency and stability. To make most of the simple heuristic described above we have to wait as long as possible before initialising a new bundle of features. This means that we deliberately wait until tracking becomes nearly unstable (because only few features are visible and possibly those features are not spread well in the image.) To increase stability, the map could be divided into fixed features and temporary features. The map is continually augmented with temporary features until a camera pose is reached where no fixed features are observable. Then temporary features are removed from the state and a new bundle of fixed features is initialised. In this way "spatial overlap" between bundles would be reduced. We sacrifice some accuracy by throwing away the temporary features. However, we predict that this effect will be outweighed by increased tracking stability.

6.7. Appearance Prediction and Measurement for Bundle Features

We have noted at the outset of this chapter that one goal of our feature model was to allow for accurate appearance prediction. In Section 6.3.3 we have hinted at the warp function $w^{\mathcal{CA}}$ that maps pixel coordinates from the reference image to the current image. In this section we derive its precise formulation for inverse depth bundle features. (Of course, the function is the same for view-point based features.) The warp function is used to create the matching template in camera $\mathcal{C}$ from the reference template in camera $\mathcal{A}$. Because we employ a stereo camera we actually have two warp functions, one for each eye. The reference template in both cases is the right (reference) image of the stereo pair from $\mathcal{A}$. The function $w^{\mathcal{CA}}$ maps to the right (reference) camera image at current pose $\mathcal{C}$ and $w^{\mathcal{C}'\mathcal{A}}$ maps to the left camera image at pose $\mathcal{C}'$. This is illustrated in Figure 6.7.

At the end of the section we give some details regarding our implementation of matching template prediction and template search in the current image.

6.7.1. Warp Function

We first derive the warp for the right eye in the following, roughly following Molton et al. (2004a). The derivation for the left eye warp is largely analogous.

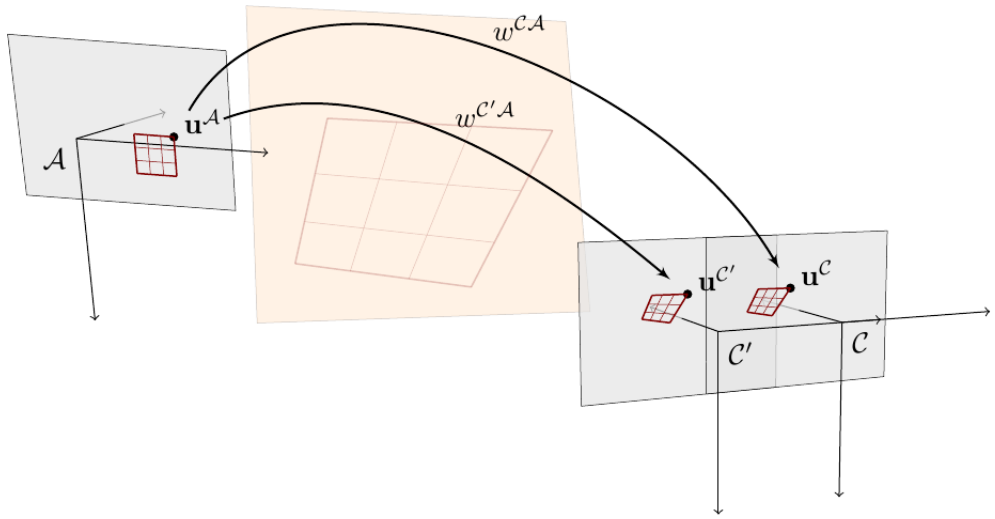


Figure 6.7.: The projections of the reference template into the images of the current stereo pair are homography transformations $w^{\mathcal{C}\mathcal{A}}$ and $w^{\mathcal{C}'\mathcal{A}}$ between the image planes.

As discussed in Section 6.3 under the planar surface assumption the warp function is a homography and depends on the poses $\mathcal{A}$ and $\mathcal{C}$, and the pose of the plane. Lets make this dependency explicit by parametrising $w^{\mathcal{C}\mathcal{A}}$ as follows

$$w^{\mathcal{C}\mathcal{A}}\left(\,\cdot\,;\,\mathbf{r}^{\mathcal{W}\mathcal{C}},\mathbf{q}^{\mathcal{W}\mathcal{C}},\mathbf{c},\phi,\rho,\mathbf{m}^{\mathcal{A}},\mathbf{n}^{\mathcal{A}}\right):\mathbb{R}^2\to\mathbb{R}^2. \tag{6.43}$$

The parameter list after the semicolon comprises the pose of the current camera ($\mathbf{r}^{\mathcal{W}\mathcal{C}}$, $\mathbf{q}^{\mathcal{W}\mathcal{C}}$) and the reference anchor pose ($\mathbf{c}$, ϕ), both with respect to the world coordinate system. The last three parameters specify the plane in the coordinate system of the anchor: ρ and $\mathbf{m}^{\mathcal{A}}$ encode a point on the plane, and $\mathbf{n}^{\mathcal{A}}$ is the normal vector of the plane.

We start by deriving a homography in normalised coordinates.[11] Consider a normalised image point in the reference image with homogeneous coordinates $\tilde{\mathbf{v}}^{\mathcal{A}}=(u,v,1)$.

In the first step, we compute the projection of $\tilde{\mathbf{v}}^{\mathcal{A}}$ onto the feature plane. We need the equation of the feature plane in the coordinate system of the anchor $\mathcal{A}$. Given the normal $\mathbf{n}^{\mathcal{A}}$ and the coordinates of any point $\mathbf{x}_0$ on a plane, we can write down the implicit equation of the plane. Every point $\mathbf{x}$ on the plane must satisfy ${\mathbf{n}^{\mathcal{A}}}^{\top}(\mathbf{x}-\mathbf{x}_0)=0$. One particular point on the plane is given by the 3D feature coordinates in the anchor coordinate system $\mathbf{x}_0=\mathbf{y}^{\mathcal{A}}=\rho^{-1}\mathbf{m}^{\mathcal{A}}$. Hence, the plane is the set of points $\mathbf{x}$ satisfying

$${\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{x}=\frac{1}{\rho}{\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{m}^{\mathcal{A}}. \tag{6.44}$$

Now we project the normalised image point $\tilde{\mathbf{v}}^{\mathcal{A}}$ onto this plane. The set of points $\{k'\tilde{\mathbf{v}}^{\mathcal{A}}\,|\,k'\in\mathbb{R}^+\}$ is the ray of all points that project to $\tilde{\mathbf{v}}^{\mathcal{A}}$ in the image plane. The

[11] Image coordinates under the identity calibration matrix $\mathsf{K}=\mathbf{I}$, i.e., the perspective projection to the plane $z=1$. Normalised coordinates were introduced in Section 3.5.

back-projection of $\tilde{\mathbf{v}}^{\mathcal{A}}$ onto the feature plane is found as the intersection of this ray with the feature plane. We substitute $\mathbf{x} = k\tilde{\mathbf{v}}^{\mathcal{A}}$ into Equation 6.44

$$k{\mathbf{n}^{\mathcal{A}}}^{\top}\tilde{\mathbf{v}}^{\mathcal{A}} = \frac{1}{\rho}{\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{m}^{\mathcal{A}} \tag{6.45}$$

and solve for k

$$k = \frac{{\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{m}^{\mathcal{A}}}{\rho{\mathbf{n}^{\mathcal{A}}}^{\top}\tilde{\mathbf{v}}^{\mathcal{A}}}. \tag{6.46}$$

We have obtained the projection $k\tilde{\mathbf{v}}^{\mathcal{A}}$ onto the plane which is represented in the anchor coordinate system $\mathcal{A}$. Next, we transform it to the current camera coordinate system, that is, we compute $p^{\mathcal{CA}}\left(k\tilde{\mathbf{v}}^{\mathcal{A}}\right)$. The transformation $p^{\mathcal{CA}}$ can be derived from the pose of the current camera $p^{\mathcal{WC}}$ and the anchor pose $p^{\mathcal{WA}}$.

$$p^{\mathcal{CA}} = p^{\mathcal{CW}} \circ p^{\mathcal{WA}} = {p^{\mathcal{WC}}}^{-1} \circ p^{\mathcal{WA}} = \left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathsf{R}_{\boldsymbol{\phi}},\ \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)\right). \tag{6.47}$$

Transforming $k\tilde{\mathbf{v}}^{\mathcal{A}}$ to the current camera yields

$$p^{\mathcal{CA}}\left(k\tilde{\mathbf{v}}^{\mathcal{A}}\right) = \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathsf{R}_{\boldsymbol{\phi}}\frac{{\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{m}^{\mathcal{A}}}{\rho{\mathbf{n}^{\mathcal{A}}}^{\top}\tilde{\mathbf{v}}^{\mathcal{A}}}\tilde{\mathbf{v}}^{\mathcal{A}} \ +\ \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right). \tag{6.48}$$

This gives the coordinates of the back-projected point in the current camera coordinate system, $\mathcal{C}$. Note that these 3D coordinates are identical to the homogeneous coordinates of the normalised image point in $\mathcal{C}$.[12] Scaling the 3D vector (the homogeneous coordinates) does not change its image projection (the non-homogeneous coordinates). We scale by $\rho{\mathbf{n}^{\mathcal{A}}}^{\top}\tilde{\mathbf{v}}^{\mathcal{A}}$ and obtain

$$\begin{aligned}
\tilde{\mathbf{v}}^{\mathcal{C}} &= p^{\mathcal{CA}}\left(k\tilde{\mathbf{v}}^{\mathcal{A}}\right)\cdot\left(\rho{\mathbf{n}^{\mathcal{A}}}^{\top}\tilde{\mathbf{v}}^{\mathcal{A}}\right) && (6.49)\\
&= \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathsf{R}_{\boldsymbol{\phi}}{\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{m}^{\mathcal{A}}\tilde{\mathbf{v}}^{\mathcal{A}} \ +\ \rho\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right){\mathbf{n}^{\mathcal{A}}}^{\top}\tilde{\mathbf{v}}^{\mathcal{A}} && (6.50)\\
&= \underbrace{\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathsf{R}_{\boldsymbol{\phi}}{\mathbf{n}^{\mathcal{A}}}^{\top}\mathbf{m}^{\mathcal{A}} \ +\ \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right){\mathbf{n}^{\mathcal{A}}}^{\top}\right)}_{\mathsf{H}_N}\tilde{\mathbf{v}}^{\mathcal{A}}. && (6.51)
\end{aligned}$$

The choice of scale factor has, on the one hand, eliminated the potential division by $\rho = 0$ in Equation 6.48. On the other hand, it allows to express $\tilde{\mathbf{v}}^{\mathcal{C}}$ as the product of a 3×3 matrix and $\tilde{\mathbf{v}}^{\mathcal{A}}$. Thus we have found the homography matrix H_N which relates normalised homogeneous coordinates in the reference and current image as $\tilde{\mathbf{v}}^{\mathcal{C}} = \mathsf{H}_N\tilde{\mathbf{v}}^{\mathcal{A}}$. The expression for H_N has been highlighted in (6.51).

Plugging in non-normalised image coordinates $\tilde{\mathbf{u}}^{\mathcal{A}} = \mathsf{K}\tilde{\mathbf{v}}^{\mathcal{A}}$ and $\tilde{\mathbf{u}}^{\mathcal{C}} = \mathsf{K}\tilde{\mathbf{v}}^{\mathcal{C}}$ we obtain

$$\tilde{\mathbf{u}}^{\mathcal{C}} = \mathsf{K}\tilde{\mathbf{v}}^{\mathcal{C}} = \mathsf{K}\mathsf{H}_N\tilde{\mathbf{v}}^{\mathcal{A}} = \underbrace{\mathsf{K}\mathsf{H}_N\mathsf{K}^{-1}}_{\mathsf{H}^{\mathcal{CA}}}\tilde{\mathbf{u}}^{\mathcal{A}} \tag{6.52}$$

and thus derive the (non-normalised) homography $\mathsf{H}^{\mathcal{CA}} = \mathsf{K}\mathsf{H}_N\mathsf{K}^{-1}$.

[12] That is, the normalised image point $\mathbf{v}^{\mathcal{C}}$ is obtained projecting to the plane $z = 1$, respectively dehomogenising $\tilde{\mathbf{v}}^{\mathcal{C}}$ (see Equation 3.21).

Finally we wrap the homography by the projection and un-projection functions (see Equations 3.21, 3.22 in Section 3.4) to get the warp function for non-homogeneous points $\mathbf{u}^{\mathcal{A}}$. In summary, we obtain

$$w^{\mathcal{CA}}\left(\,\cdot\,;\mathbf{r}^{\mathcal{WC}},\mathbf{q}^{\mathcal{WC}},\mathbf{c},\boldsymbol{\phi},\rho,\mathbf{m}^{\mathcal{A}},\mathbf{n}^{\mathcal{A}}\right)\;:\;\mathbf{u}^{\mathcal{A}}\mapsto h\left(\mathsf{H}^{\mathcal{CA}}h^{-1}\left(\mathbf{u}^{\mathcal{A}}\right)\right). \tag{6.53}$$

with

$$\mathsf{H}^{\mathcal{CA}}=\mathsf{K}\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\mathsf{R}_{\boldsymbol{\phi}}\mathbf{n}^{\mathcal{A}\top}\mathbf{m}^{\mathcal{A}}\;+\;\rho\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}^{\mathcal{A}\top}\right)\mathsf{K}^{-1}. \tag{6.54}$$

This is the warp function for the right eye of the stereo camera.

Warp function for the left eye For the left eye, the derivation is analogous. Instead of $p^{\mathcal{CA}}$ we use $p^{\mathcal{C}'\mathcal{A}}$ to project from the plane onto the left image plane. From the relative pose of the camera eyes which is a shift along the baseline

$$p^{\mathcal{C}'\mathcal{C}}=(\mathbf{I},\;\mathbf{b})\;\text{ with }\mathbf{b}=(b,0,0)^{\top} \tag{6.55}$$

we can derive

$$p^{\mathcal{C}'\mathcal{A}}=p^{\mathcal{C}'\mathcal{C}}\circ p^{\mathcal{CA}}=\left(\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathsf{R}_{\boldsymbol{\phi}},\;\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)+\mathbf{b}\right). \tag{6.56}$$

Going through the derivation analogous to Equations 6.48 - 6.52 we find that that the baseline shift shows up as an additive term in the homography matrix. We obtain

$$w^{\mathcal{C}'\mathcal{A}}\left(\,\cdot\,;\mathbf{r}^{\mathcal{WC}},\mathbf{q}^{\mathcal{WC}},\mathbf{c},\boldsymbol{\phi},\rho,\mathbf{m}^{\mathcal{A}},\mathbf{n}^{\mathcal{A}}\right)\;:\;\mathbf{u}^{\mathcal{A}}\mapsto h\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}h^{-1}\left(\mathbf{u}^{\mathcal{A}}\right)\right). \tag{6.57}$$

with

$$\mathsf{H}^{\mathcal{C}'\mathcal{A}}=\mathsf{H}^{\mathcal{CA}}+\mathsf{K}\left(\rho\mathbf{b}\mathbf{n}^{\mathcal{A}\top}\right)\mathsf{K}^{-1} \tag{6.58}$$

6.7.2. Implementation Details of the Measurement Process

In this section we want to give a brief overview of our implementation of the measurement process for bundle (or view-point based) features.

To measure a feature $\mathbf{y}_{j,i}$, we compute the estimated warp functions for the current left and right image to create matching templates for the feature. To obtain the warp functions, we need the current state estimates of the anchor and camera pose as well as the feature's inverse depth. We also need the (fixed) feature ray $\mathbf{m}^{\mathcal{A}}_{j,i}$ and the feature normal. We assume that the feature was facing the camera when it was initialised, thus we use the normal vector $\mathbf{n}^{\mathcal{A}}=-\mathbf{m}^{\mathcal{A}}_{j,i}$. We obtain the estimated warp function

$$\overline{w}^{\mathcal{CA}}(\cdot)=w^{\mathcal{CA}}\left(\cdot\,;\overline{\boldsymbol{\mu}}_{\mathbf{r}^{\mathcal{WC}}},\overline{\boldsymbol{\mu}}_{\mathbf{q}^{\mathcal{WC}}},\overline{\boldsymbol{\mu}}_{\mathbf{c}_j},\overline{\boldsymbol{\mu}}_{\boldsymbol{\phi}_j},\overline{\boldsymbol{\mu}}_{\rho_{j,i}},\mathbf{m}^{\mathcal{A}}_{j,i},-\mathbf{m}^{\mathcal{A}}_{j,i}\right) \tag{6.59}$$

for the right image.

To create the matching template, we first apply the warp function to transform the outline of the reference template to the current image. In our implementation, the reference template is always a 21×21 pixel square in the reference image. Thus, we transform the corner points of the square through $\overline{w}^{\mathcal{CA}}$ to obtain a quadrilateral outline in the current

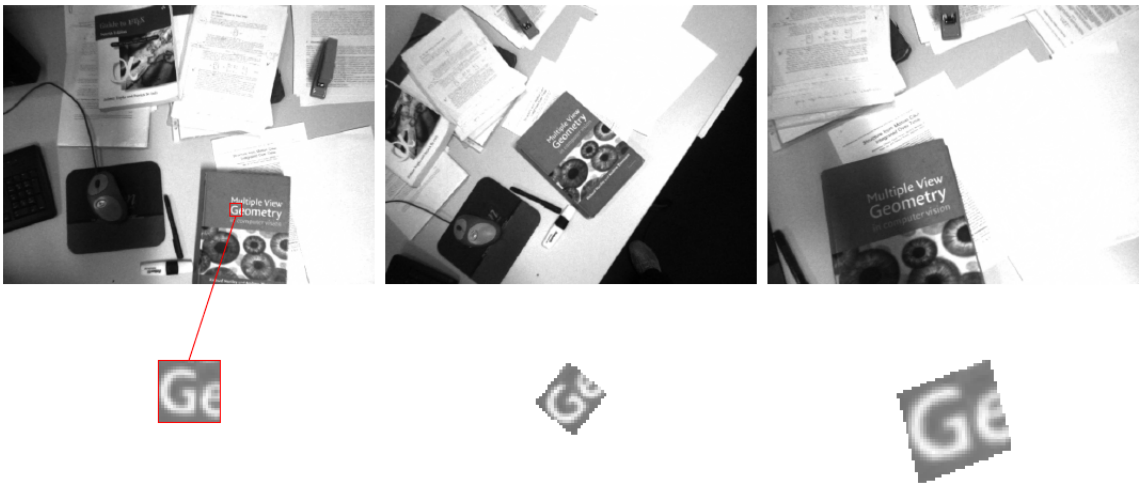


Figure 6.8.: Examples of matching template prediction for bundle features in a real image sequence. In the rightmost column, we show a reference (anchor) image and the reference template of one feature that was initialised in this image. The two remaining columns show other images of the sequence on the top, and the predicted feature appearance (matching template) below.

image. We then fill in the pixel values within this outline by bilinear sampling of the reference template using the inverse of the warp function, $\overline{w}^{\mathcal{CA}^{-1}}$. Examples are shown in Figure 6.8.

Then we scan the matching template over the predicted search region and find the pixel position with the minimum zero-mean sum of squared differences. Assume, the minimum occurs when the matching template is shifted by an integer pixel offset $\mathbf{p}$ from it's predicted position. The measurement in the right image then is the predicted image position of the feature point, shifted by $\mathbf{p}$. Thus, the measurement for the right image is

$$\mathbf{z}_r = \mathbf{h}_r + \mathbf{p} \tag{6.60}$$

where $\mathbf{h}_r$ is the predicted measurement in the right image, i.e., the u, v components of $\mathbf{h}\left(\overline{\boldsymbol{\mu}}, \mathbf{0}\right)$.

We proceed analogously for the left image. To speed up the search, we further restrict the search region in the left image to the epipolar line of $\mathbf{z}_r$.

Finally, the u, v components of the stereo measurement $\mathbf{z}$ are taken from the right image measurement $\mathbf{z}_r$, the disparity d is computed as the difference of u' of the left and u of the right measurement.

6.8. Biased Measurements

6.8.1. Appearance Bias

A subtle issue of the measurement mechanism is bias introduced by the matching template. We have argued that the ray to the feature in the anchor frame is not a measurement but

an arbitrarily chosen fixed variable. The ray direction is determined exactly and the remaining estimation problem is to find the depth of the scene along this direction. To enable future measurements, an appearance template is taken from the reference (anchor) image. The appearance template is a small patch of pixels copied from the image around the point where the feature ray intersects the image plane.

There is no uncertainty in the position of this patch. However, there is image noise involved. Pixel intensities in the reference image are affected by noise. Thus, if we sample appearance templates at the same position from several images taken from exactly the same camera pose, we will get a slightly different template every time. The template patch we take from the reference image is a noisy version of the ideal noise-free patch. We match against this template in all subsequent measurements. Because the intensity noise in the template is the same for all measurements this may cause a bias in the measurements. For instance, if by some unfortunate noise pattern the template we take looks like the ideal noise-free template shifted by one pixel to the right, then there is a one pixel bias in all subsequent measurements. We will label this effect *appearance bias* in the following.

The Kalman Filter is based on the assumption that measurement noise is zero-mean. Any systematic bias in the measurements is potentially harmful.

Aside from the anchor pose parameters, our view-point based and inverse depth bundle models only have one parameter per feature. Appearance bias is not modeled and hence assumed to be zero. Other models add 2D bias parameters which effectively allow to shift the observed template in the image plane. For instance, the 3-parameter Euclidean model allows the position of the feature to vary not only along a fixed ray but also perpendicular to this ray. As noted by Azarbayejani & Pentland (1995) this is analytically equivalent to adding 2D bias parameters to a 1-parameter model.

These considerations raise several questions. Is it harmful to ignore appearance bias? What is the benefit of adding 2D bias parameters to the model? Is a 2D bias enough to adequately subsume the appearance bias which is in fact of much higher dimension (one bias parameter per template pixel).

In the literature we can find arguments *pro* and *contra* bias parameters. Azarbayejani & Pentland (1995) propose a 1-parameter model similar to ours. However they use depth instead of inverse depth and there is only a single "anchor" respectively reference frame (which is fixed at the first frame of the sequence and is not explicitly represented). With respect to measurement bias they note that "it is common to use Kalman filters even when measurements are not truly zero-mean. Good results can be obtained if the biases are small." They also point out that "there is a trade-off between the accuracy that might be gained by estimating bias and the stability of the filter, which is reduced when the state vector is enlarged" (Azarbayejani & Pentland, 1995, p. 566). They perform simulation experiments with artificially biased measurements and find that even large biases only have moderate effect on accuracy. However, they do not present comparative experiments to a model including bias parameters.

This 1-parameter model is criticised by other authors. McLauchlan & Murray (1995) propose the Variable State Dimension Filter, which is basically an EKF with a batch initialisation stage. In their experiments they present a comparison of a 1-parameter and a 3-parameter model on simulated data. They find that with respect to camera motion estimation the 1-parameter model is comparable to the 3-parameter model. However,

comparing the recovered map for both models the 1-parameter model "rapidly reaches the point where it can no longer improve its estimates, because the remaining errors are in the fixed rays computed from the first image" (McLauchlan & Murray, 1995, p. 319).

Chiuso et al. (2002) argue that 1-parameter models are *sub-minimal* because they fail to capture the appearance bias.[13] Their experiments include a comparison with a sub-minimal model which is shown to catastrophically fail on longer sequences (after approximately 200 frames), showing very large errors in the estimated camera pose. The authors note that "this effect is not visible on short sequences, which is probably why it has not been noticed by Azarbayejani and Pentland" (Chiuso et al., 2002, p. 526). However, another explanation is more likely. Azarbayejani & Pentland (1995) only initialise features in the very first frame of the sequence and therefore only need a single reference frame. Chiuso et al. (2002) extend this method by adding new features at later points of the sequence. However, there still is only a single reference frame, namely the first frame of the sequence. New features are transferred back to this initial frame. Then, of course there will be a strong bias in the fixed rays. However, to the overwhelmingly large part, this bias is due to the error in the current camera pose estimate. It is correct that this error should not be neglected. In our model, this is accomplished by adding a new anchor to the state which correctly captures the uncertainty in the current camera pose. In the model of Chiuso et al. (2002) this error must be handled by the bias parameters. These parameters therefore do much more than capture appearance bias. Removing the parameters causes the catastrophic failure that the authors observe.

A critical point in the experiments by McLauchlan & Murray (1995) and Chiuso et al. (2002) is that they use point cloud simulations where the generation of measurements is oversimplified. Measurements are generated by computing the ideal image projection of a point and adding Gaussian measurement noise. The first "measurement", too, is generated in this way. Thus, the simulation is constructed with the same view of the measurement mechanism that gives rise to the 3-parameter models. The simulation is tailored after this model. It is quite obvious that the experiments give favourable results for the 3-parameter feature models in this case.

However, as we have pointed out, the initial observation is *not* obtained in the same way as the others and is *not* influenced by the same noise. Taking this point of view, in a point cloud simulation we would not add noise to the initial observation. In this case, we could clearly achieve results that favour the 1-parameter feature model.

Undeniably, neither point of view captures the true generative process in full detail. A fair comparison of the 1-parameter and 3-parameter model is not possible in this way.

6.8.2. Position-Dependent Bias

We have noted initially that there *is* appearance bias in the initial model. It is not clear whether this can be subsumed in 2D bias parameters. Still, in the absence of other biases

[13] It is noteworthy that the same authors have no qualms for eliminating the bias parameters in later work (Jin et al., 2003). In this work they model features explicitly as planar surface patches. The level of abstraction of the measurement process is reduced, moving the focus much closer to the actual pixel intensities. It seems that the closer one looks at what underlies the generic measurements, the less intuitive it becomes to treat initialisation as "just another measurement."

we would expect a 3-parameter model to perform slightly better than a 1-parameter model. However, we have to take into consideration other sources of measurement bias such as

- non-Lambertian surface reflectance,
- incorrectly assumed surface normal, and
- non-planar scene surfaces (e.g., features on occlusion boundaries).

These will cause unmodeled appearance changes when the feature is observed from different view points. We will subsume these effects as *position-dependent bias*.

Position-dependent biases are captured by neither the 1-parameter nor the 3-parameter models. There is the danger that a 3-parameter model uses the additional degrees of freedom to erroneously adapt to position-dependent variations and settles on a biased estimate after the first few observations. In this way, the simplification introduced by a 1-parameter model might actually have a stabilising effect.

6.8.3. Adding Bias Parameters to the Inverse Depth Bundle Model

The SLAMDUNK evaluation framework presented in Chapter 5 provides the means to compare different models on data that is more realistic than point cloud simulation. We identified the following issues for investigation:

- Can we observe catastrophic failure on long sequences similar to Chiuso et al. (2002) for the 1-parameter inverse depth bundle model?
- What is the influence of appearance bias on the accuracy of the reconstructed map and the reconstructed trajectory? We can evaluated this by making the appearance bias the only source of bias. That is, the models should be evaluated on a sequence where all features lie on planar Lambertian surfaces with known orientation.
- How do the models behave under more realistic conditions? How do appearance and position-dependent bias interact? This would require evaluation on a sequence including non-planar features and/or unknown feature orientation.

To assess the 3-parameter model type, we add bias parameters to the 1-parameter bundle features. A biased feature $\mathbf{y}_{j,i}$ with respect to anchor $\mathbf{a}_j$ is parameterised as

$$\mathbf{y}_{j,i} = \begin{pmatrix} \rho_{j,i} \\ \Delta u_{j,i} \\ \Delta v_{j,i} \end{pmatrix} \tag{6.61}$$

The bias parameters $\mathbf{d}_{j,i} = (\Delta u_{j,i}, \Delta v_{j,i})$ describe a shift of the observed initial template on the image plane. This shift allows for the partial adaption to appearance bias. The same could be encoded by allowing the physical scene point to move away from the feature ray. We have chosen the direct image plane bias parameterisation for two reasons. First, it parameterises bias along axes physically relevant to the measurement process. Second, allowing motion away from the feature ray would favour the 1-parameter model in the

experimental evaluation because ground truth feature positions are generated by tracing along the fixed initial ray.

In the measurement process, the shift in the template is realised by post-multiplying the warping homography $\mathsf{H}^{\mathcal{CA}}$ (see Section 6.7) with a shifting transformation

$$\mathsf{H}^{\mathcal{A}\widetilde{\mathcal{A}}} = \begin{bmatrix} 1 & 0 & \Delta u_{j,i} \\ 0 & 1 & \Delta v_{j,i} \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.62}$$

The generative measurement model (6.34) must also be adapted such that the bias update through measurements is correctly handled. The predicted position in the current image is obtained by adding the bias to the feature center in the image plane, $\pi(\mathbf{m}^{\mathcal{A}}_{j,i})$, back-projecting to the feature plane and projecting to the current image. We do not present the details of the measurement model here, as the derivation proceeds largely analogously to the derivation of the view-point based measurement model in Section 6.5.1. An experimental comparison of the original bundle model and this biased version is part of the next section.

6.9. Experimental Evaluation

The goal of this section is to assess the inverse depth bundle model in the context of a practical visual SLAM system. We expect that the model will be a more efficient representation in terms of state size than other parameterisations. This is caused by sharing the anchor information among a bundle of features. Moreover, we use a one-parameter model for each feature with respect to its anchor. In contrast to other parameterisations the feature model does not include further parameters that enable an adaption to measurement bias. This might cause a decline in accuracy. In the worst case, we might even see divergence in the EKF and consequential failure. However, the enforced rigidness of the bundle model might provide additional stability against non-appearance bias influences. In Section 6.9.1 we investigate the implications of using the one-parameter bundle model instead of a model comprising bias parameters. This is accomplished using rendered image sequences with known ground truth. In Section 6.9.2 we investigate the benefit of the bundle model in terms of state size reduction. For this purpose we use real image sequences of typical indoor and outdoor scenarios.

6.9.1. Analysis of Bias Effects

Evaluation Procedure

We compare the accuracy of the bundle model and a model including bias parameters on rendered image sequences. To distinguish between pure appearance bias and measurement bias in general, we analyse two scenarios.

The first scenario concerns appearance bias only. We eliminate other sources of bias and try to create ideal conditions where all assumptions are satisfied (except, of course,

the assumption of non-existent appearance bias for the bundle model.) We use scenes consisting of Lambertian planar surfaces with known orientation and constant illumination. Gaussian noise is added to the rendered images to evoke appearance bias.

In the second scenario we move towards more realistic conditions. The scene consists of several planar segments. Now, the surface orientations are unknown, which causes position-dependent bias. Moreover, the rendered sequence is motion-blurred which introduces another source of bias.

The two parameterisations to be compared are

- **"1-param"** This is the inverse depth bundle model described in Section 6.6, i.e., a parameterisation where the feature has only one degree of freedom with respect to its anchor camera.
- **"biased"** This is the bundle model with bias parameters described in Section 6.8.3. This is a representative of the class of feature models that allow full 3 degrees of freedom per feature. The unified inverse depth parameterisation by Montiel et al. (2006) and the Euclidean parameterisation both fall within this class. Specifically, here, the bundle model is augmented with two bias parameters per feature, which allow a shift of the feature template in the image plane. In this model the bias is directly modeled after the measurement process, which provides a high quality feature model of this class.

Additionally, we evaluate both models with a pixel-accurate and a subpixel-accurate feature matching method.

For the experiments, we implement the models within the visual SLAM framework described in Chapter 4. Except for the feature model, the systems used for evaluation are identical. To ensure a fair comparison, all visual SLAM runs use the same pre-recorded feature initialisations. Furthermore, all visible features are measured in every frame. The intention is that the same features measurements are attempted in every run. The pre-recorded list of feature initialisations is constructed in the following way: The visual SLAM system is run on the sequence using the normal feature selection and initialisation process. The initialisation events are logged. The SLAMDUNK evaluation framework is used to augment every initialisation event by the ground truth feature normal. Features that cause mismatches due to repetitive or insufficient texture should be removed from the initialisation list because they could corrupt the results of the experiment. Thus, we remove from the initialisation list features for which outlier measurement errors of 5 pixels or more occur during the recording run.

For each run, the camera estimate of the visual SLAM system is initialised to the ground-truth pose with no uncertainty.

The accuracy of both models on the test sequences is evaluated using the SLAMDUNK framework. The following evaluation measures are used.

- **Absolute camera position error.** The absolute camera position error is defined as the Euclidean distance between the ground truth and the estimated camera pose. It is computed for each frame of the sequence and plotted over time. Besides the absolute error, this plot also gives an indication of the jitter in the reconstructed trajectory.

- **Best-fit map error.** We define the map error as the root mean squared distance between estimated map features and corresponding ground truth features. We compute a best-fit rigid transformation of the estimated map such that the map error is minimized. The best-fit map error is defined as the residual map error after applying this best-fit transformation. It is computed for each frame of the sequence and plotted over time. The best-fit map error is used to evaluate the accuracy of the map in itself at every frame. In particular, this quantity is neither influenced by a misalignment of the map with respect to the world coordinate frame, nor by the current camera pose error.

- **Overall measurement error.** For every feature, in every frame, the perfect measurement is the ground truth stereo projection of the feature center into the ground truth camera. The absolute measurement error is defined as the distance (in pixels) between the actual feature measurement and the perfect measurement. To assess the overall measurement quality, we compute the mean and standard deviation of the absolute measurement error over all measurements of a sequence.

- **Per feature measurement error distribution.** For selected features we plot the 2D distribution of their measurement error in the reference image over the sequence. Here, the measurement error is defined as the 2D offset (in pixels) between the actual feature measurement and the perfect measurement. Bias effects are clearly visible in these scatter plots.

- **Per feature measurement bias.** For every feature, we estimate the measurement bias in the reference image in the following way. For every measurement of the feature, we compute the measurement error, i.e., the 2D offset (in pixels) between the actual measurement and the perfect measurement. The magnitude of the mean of these 2D offsets is the measurement bias for this feature.

Appearance Bias Only

We begin with a very simple sequence which is designed to eliminate to the best possible degree sources of error other than appearance bias.

"Plane" Sequence. Figure 6.9 shows a schematic of the "plane" scene and trajectory. The scene comprises a 5 m × 3 m plane at 2 m distance. The plane is texture-mapped with a photograph. The camera is modeled after a Point Grey Bumblebee® stereo camera with 640 × 480 resolution and 65° horizontal field of view. The camera moves along an elliptic trajectory, with a major radius of 1.5 m and a minor radius of 0.5 m, centered at the origin. Throughout the trajectory the camera fixates a point 3 m behind the plane. At the start of the sequence, the camera is stationary for the first 10 frames. Then, it slowly accelerates. The initial stationary phase is intended to allow the bias parameters of the "biased" model to adapt to the additive image noise. In this way, we avoid that the bias parameters by mistake adapt to the image motion.

Figure 6.10 shows selected images from the rendered "plane" sequence. As we have discussed in Section 6.8.1, appearance bias is caused by image noise. We run experiments

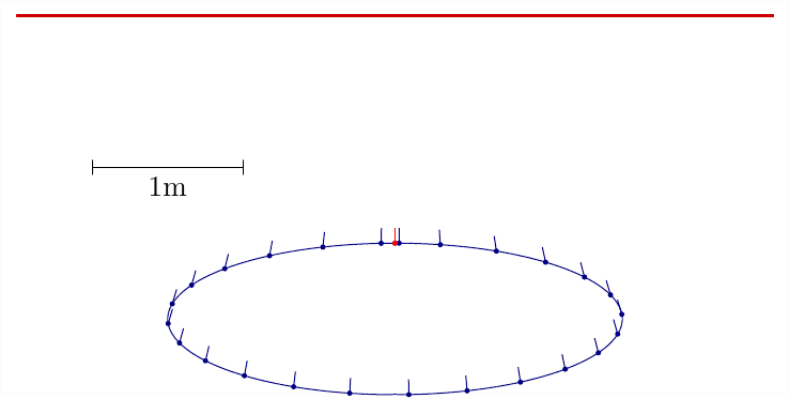


Figure 6.9.: Schematic top view of the "plane" sequence. The camera moves along an elliptic trajectory (blue) in front of a textured plane (red). For every 15th frame, the cameras viewing direction is indicated by a vector in the direction of the optical axis.

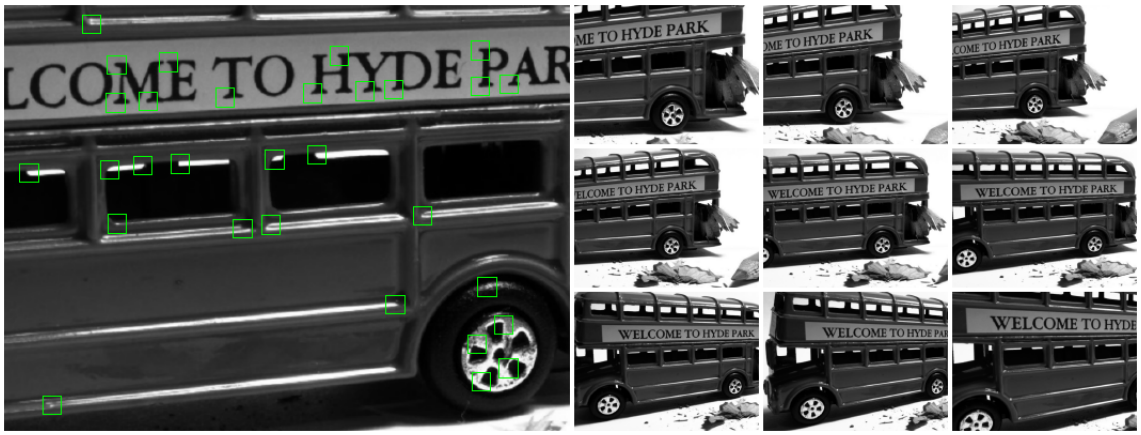
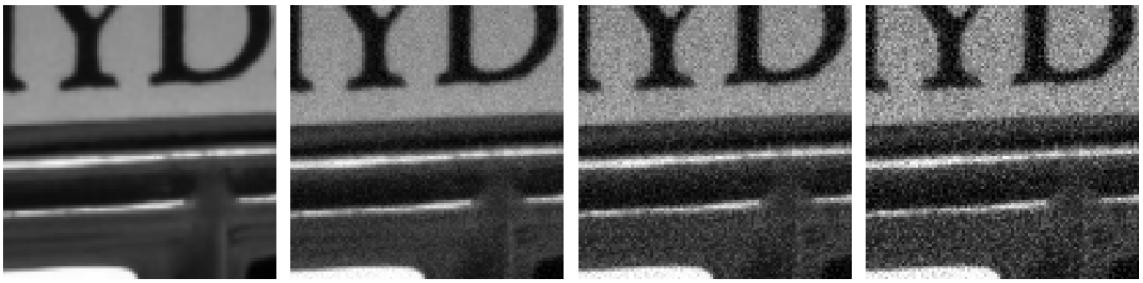


Figure 6.10.: Selected images of the "plane" sequence. On the left, the first image of the sequence is shown with the selected features overlaid.



$\sigma = 0$ $\sigma = 10$ $\sigma = 20$ $\sigma = 30$

Figure 6.11.: A fragment of the initial image of the "plane" sequence with different levels of additive noise.

with varying amount of noise added to the rendered images. We use per-pixel independent additive Gaussian noise with standard deviation ranging from $\sigma = 0$ (no noise) to $\sigma = 30$ intensity levels.[14] We run experiments for four settings of noise which are illustrated in Figure 6.11. We expect that the appearance bias will increase with higher noise levels.

A comparison of the absolute camera position errors achieved by the "1-param" and "bias" models is shown in Figure 6.12. The error is mostly below 20 mm, for both models and all noise settings. It can be seen that increased additive noise mainly increases jitter in the estimated trajectory. With respect to the position error, the models behave similarly. The difference of errors between the models is below the level of jitter.

The comparison of the best-fit map errors in Figure 6.13 is more informative. For moderate noise levels the "1-param" model performs similar to or even better than the "biased" model. The map error of the "1-param" model consistently converges to lower values towards the end of the sequence. Surprisingly, this also happens for the experiment with $\sigma = 0$, i.e., without additive noise. A possible explanation is that there are small position-dependent biases introduced by aliasing in the ray-tracer on which the "biased" model picks up. The more rigid "1-param" model has a stabilising effect here. Eventually, with more severe noise with $\sigma = 30$ we see the expected outcome: The map accuracy for the "1-param" model degrades because of appearance bias effects, whereas the "biased" model can partially compensate for this.

This picture is affirmed by analysing the distribution of measurement error. Mean and standard deviation of the absolute error over all measurements are illustrated in Figure 6.14. For moderate noise levels the "biased" model seems to adapt to residual position-dependent biases which leads to slightly increased measurement error. For $\sigma = 30$, appearance bias has a stronger effect on the "1-param" model. The "biased" model can partially compensate this and achieves slightly lower measurement error.

When we analyse the per feature measurement error distribution we find that the biases in general are very small, i.e., 1/10 pixel or less. Examples are shown in Figure 6.15. We observe that for some features the "biased" model can reduce the average measurement bias, e.g., feature #28 in Figure 6.15. However, for most features the "biased" model does not reduce or even increases bias, e.g., features #54 and #57. The following table gives a

[14] The intensity range of the images is $[0, 255]$.

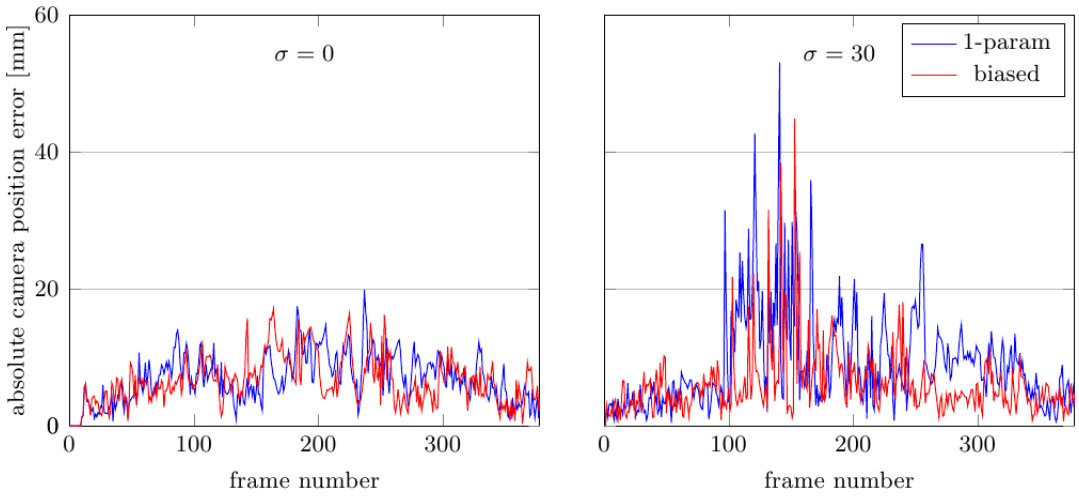


Figure 6.12.: Absolute camera position error for the "plane" sequence. The left plot shows the results for the sequence without additive noise. The right plot shows the results for additive Gaussian noise with standard deviation of $\sigma = 30$ intensity levels.

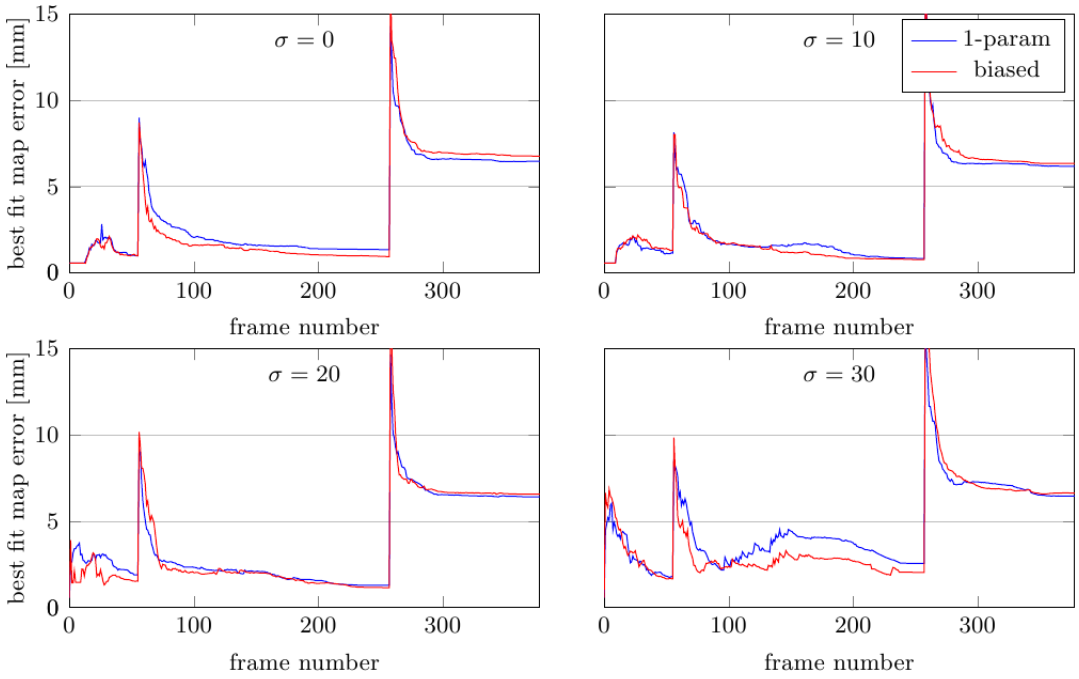


Figure 6.13.: Comparison of best-fit map error for the estimated maps for the "plane" sequence. Each subplot shows the result for one of the additive noise σ settings.

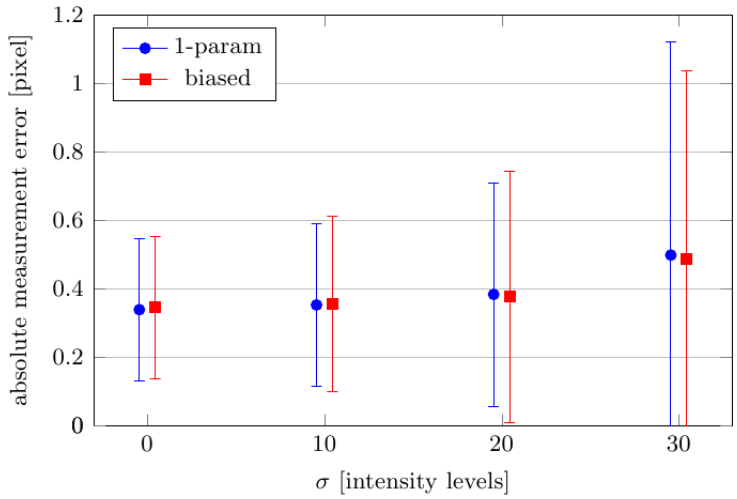


Figure 6.14.: Measurement error for the "plane" sequence. Mean and standard deviation of the absolute measurement error for the "1-param" and "biased" models are compared at different additive noise levels.

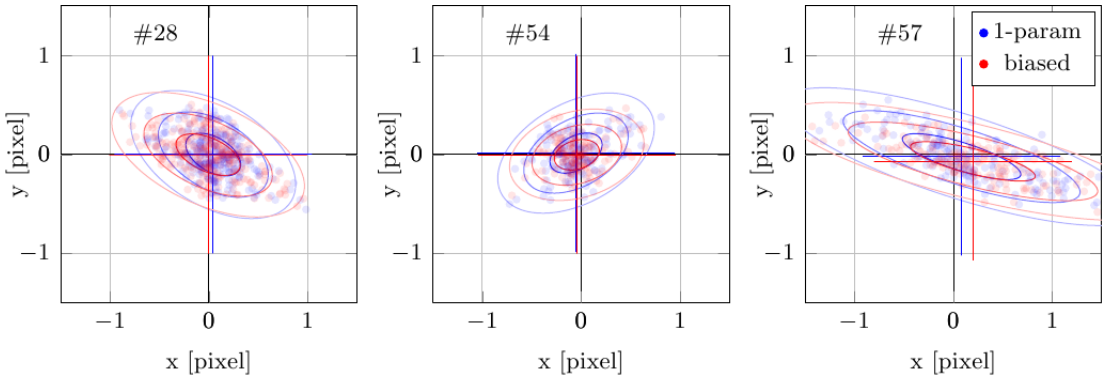


Figure 6.15.: Measurement error for selected features for the "plane" sequence. The plots show the experiment with additive Gaussian noise with $\sigma = 20$ intensity levels. The three plots show, for one feature each, the distribution of measurement error. The red and blue dots show individual measurements for the "1-param" and "biased" models, respectively. The cross-hairs indicate the mean and the ellipses the regions of 1, 2, and 3σ standard deviations.

comparison of the models with respect to the fraction of features for which the bias could be decreased. The first column gives the standard deviation of the additive noise in the sequence. The second column gives the percentage of features that have a smaller bias using the "biased" model. The third column give the percentage of features that have a smaller bias using the "1-param" model.

σ	"biased"	"1-param"
0	43%	57%
10	47%	53%
20	50%	50%
30	50%	50%

We observe that contrary to our expectations the "1-param" model achieves better bias values for 50% or more of the features, even for significant levels of noise.

"Plane" Sequence with Subpixel Matching. The results of the previous experiment suggest that the bundle model actually performs better than a model including bias parameters. This is somewhat surprising, especially because the sequence contains *only* appearance bias. A possible explanation for the inefficacy of the "biased" model in this case is that the pixel-accurate matching method only gives integer measurement coordinates. This may constitute another unmodeled bias influence which distracts the "biased" model. Therefore, we repeat the previous set of experiments, this time using subpixel refinement as described in Section 4.7.1.

Looking at Figure 6.16, we see that position error as well as camera jitter is greatly reduced by the subpixel matching. We now observe slightly smaller position errors for the biased model.

The best-fit map error, cf. Figure 6.18, is decreased for both models in comparison to the pixel-accurate matching experiment. The map error for the "1-param" and "biased" is almost indistinguishable. A very slight advantage of the "biased" model can be observed, which increases as expected with increasing noise standard deviation.

An analysis of the distribution of measurement errors shows consistently better results for the "biased" model, cf. Figure 6.18.

The majority of per feature measurement biases is improved by the "biased" model. The following table gives a comparison with respect to the fraction of features for which the bias could be decreased. Again, the second and third column give the percentage of features that have a smaller bias using the "biased" and "1-param" models, respectively.

σ	"biased"	"1-param"
0	74%	26%
10	89%	11%
20	92%	8%
30	85%	15%

Examples of per feature measurement error distribution are given in Figure 6.19.

Altogether, this second set of experiments exhibits the expected behaviour. The "biased" model successfully counters appearance bias and the advantage grows with the amount of noise in the images.

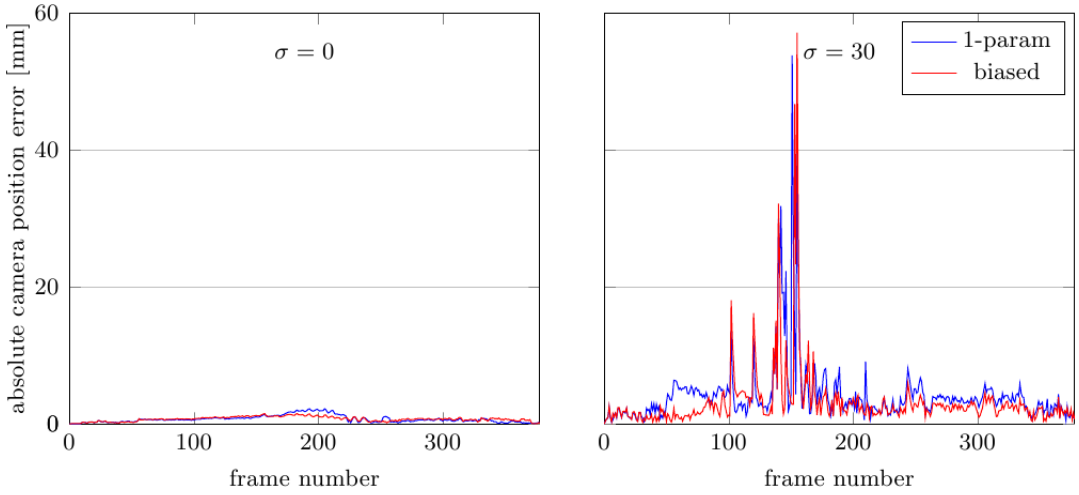


Figure 6.16.: Absolute camera position error for the "plane" sequence employing subpixel matching. The left plot shows the results for the sequence without additive noise. The right plot shows the results for additive Gaussian noise with standard deviation of $\sigma = 30$ intensity levels.

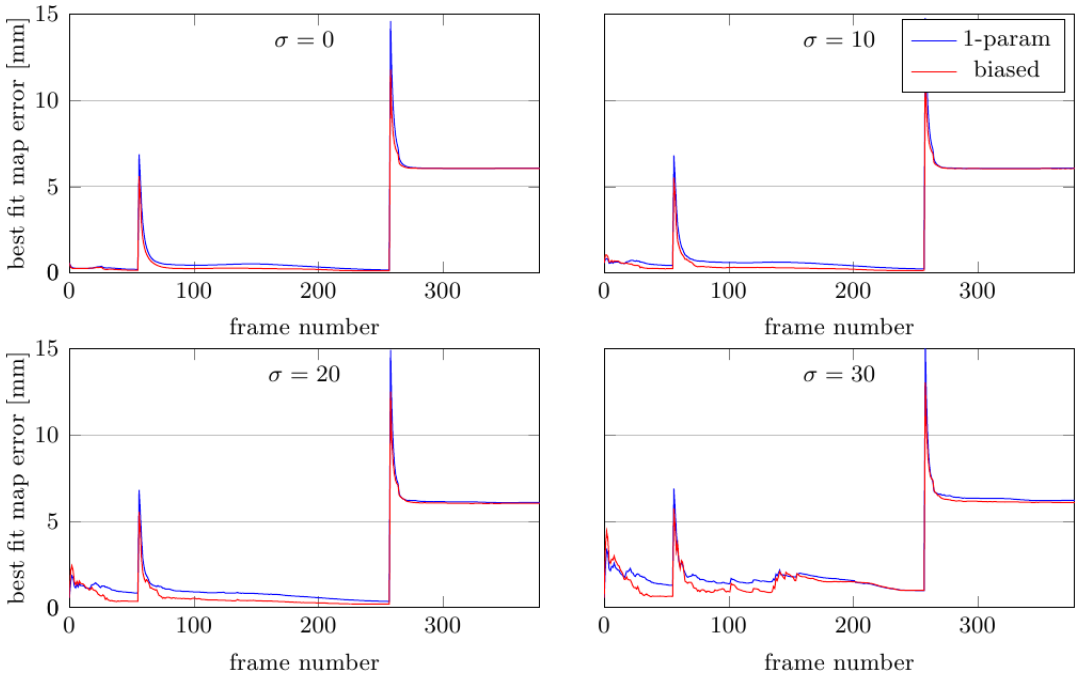


Figure 6.17.: Comparison of best-fit map error for the "plane" sequence employing subpixel matching. Each subplot shows the result for one of the additive noise σ settings.

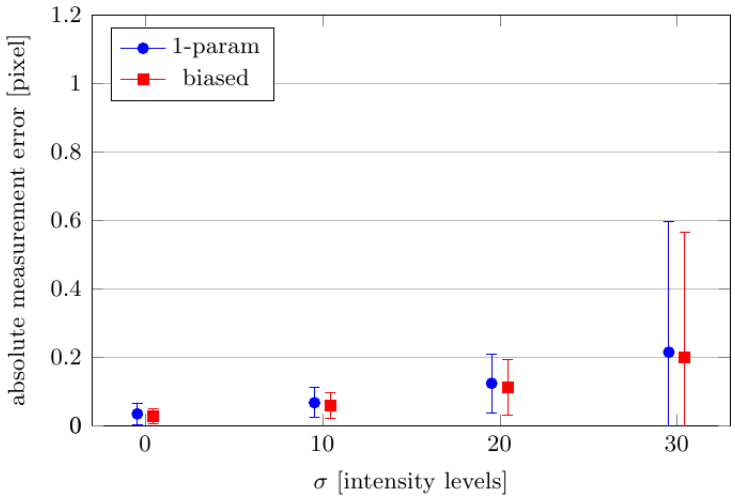


Figure 6.18.: Measurement error for the "plane" sequence employing subpixel matching. Mean and standard deviation of the absolute measurement error for the "1-param" and "biased" models are compared at different additive noise levels.

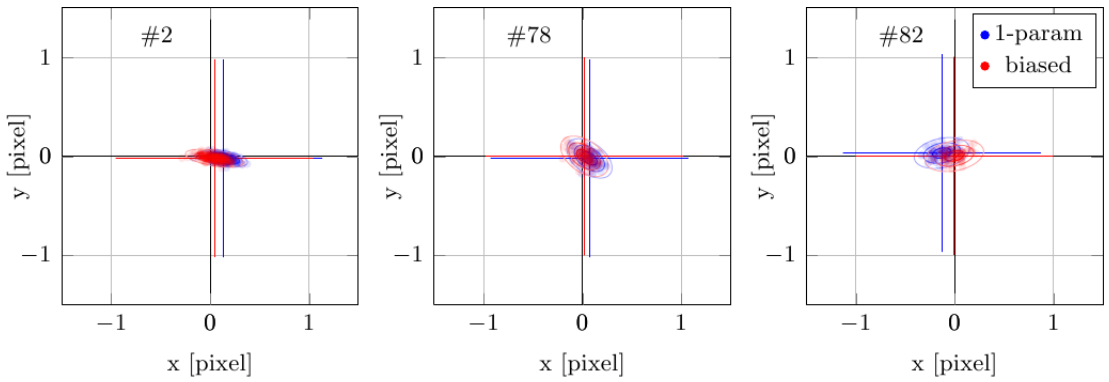


Figure 6.19.: Measurement error for selected features for the "plane" sequence employing subpixel matching. "Plane" sequence with additive Gaussian noise with $\sigma = 20$ intensity levels.

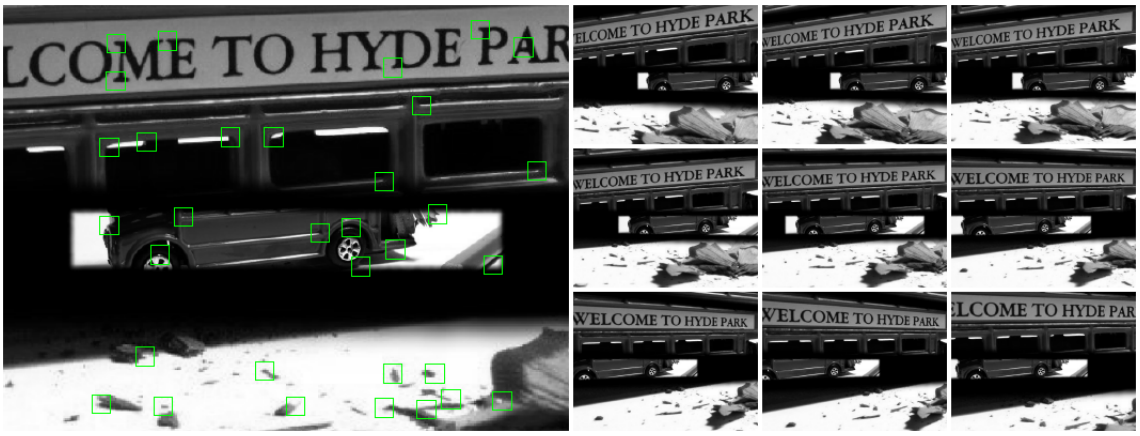


Figure 6.20.: Selected images of the "stripes" sequence. On the left, the first image of the sequence is shown with the selected features overlaid.

"Stripes" Sequence. In the "plane" sequence, all features lie close to the camera. With the second set of experiments we analyse whether the results can be confirmed for very distant features, as well. We use the "stripes" sequence for this purpose. The camera trajectory for the "stripes" sequence is the same elliptic trajectory as described for the "plane" sequence. The planar scene from the previous experiment is replaced now by several texture-mapped stripes at 3.5, 2.25, and 102 m distance respectively. These stripes can be seen in Figure 6.20, which shows some images of the sequence.

Similar to the previous experiments we evaluate with four additive noise settings and with pixel-accurate and subpixel-accurate matching. Altogether, the results of the "plane" sequence are confirmed: For pixel-accurate matching, the results of the "1-param" model are equivalent or slightly better than for the "biased" model. For subpixel matching, the "biased" model is slightly better and the improvement increases with increasing noise level. In the following, we present the results for the subpixel case in more detail.

The absolute camera position error is shown in Figure 6.21. In comparison to the "plane" sequence the distant features provide greater stability of the estimated trajectory against image noise.

The "biased" model slightly reduces measurement error in this experiment as well, cf. Figure 6.22.

The plots of the best-fit map error are especially interesting, cf. Figure 6.23. Of course, the error is larger now because the depth estimates of the distant features are less accurate. The "biased" model is better here as well. The new observation is that here the difference between the models lies mainly in speed of convergence. The map error for the "1-param" bundle model exhibits slower convergence. However, eventually it converges to similar accuracy as the "biased" model. A possible explanation for this behaviour is a stabilising effect of the rigid bundle configuration: Because we observe many features of the bundle simultaneously, measurement errors in individual features tend to cancel out.

From the previous experiments we can conclude that, when common simplifying assumptions about the scene are satisfied and when subpixel matching is used, the "bi-

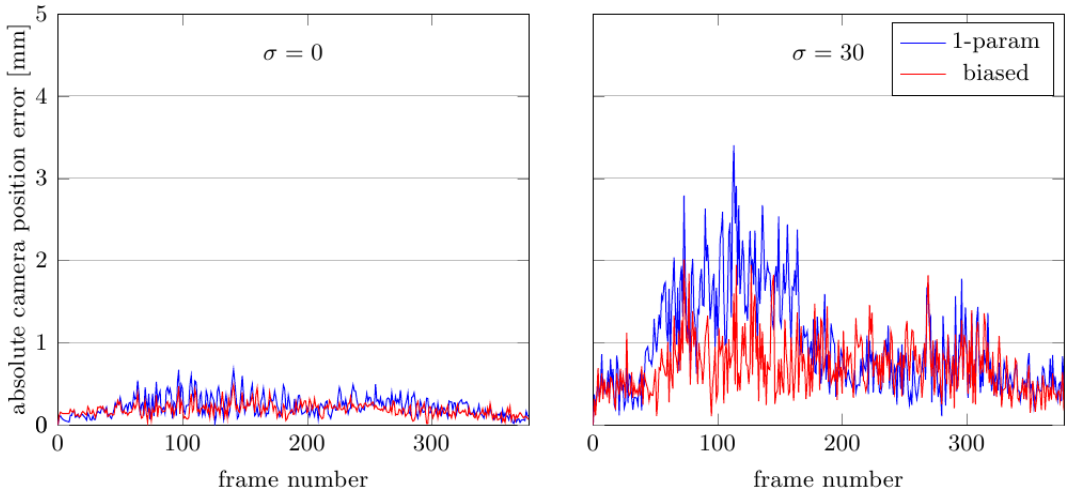


Figure 6.21.: Absolute camera position error for the "stripes" sequence employing subpixel matching. The left plot shows the results for the sequence without additive noise. The right plot shows the results for additive Gaussian noise with standard deviation of $\sigma = 30$ intensity levels.

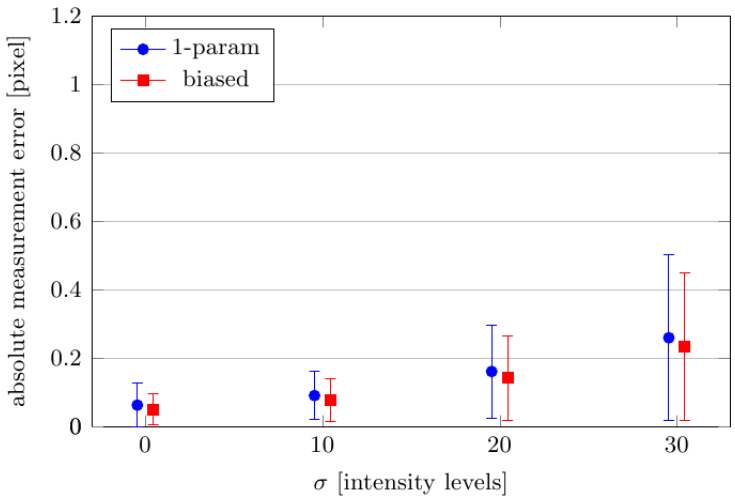


Figure 6.22.: Measurement error for the "stripes" sequence employing subpixel matching. Mean and standard deviation of the absolute measurement error for the "1-param" and "biased" models are compared at different additive noise levels.

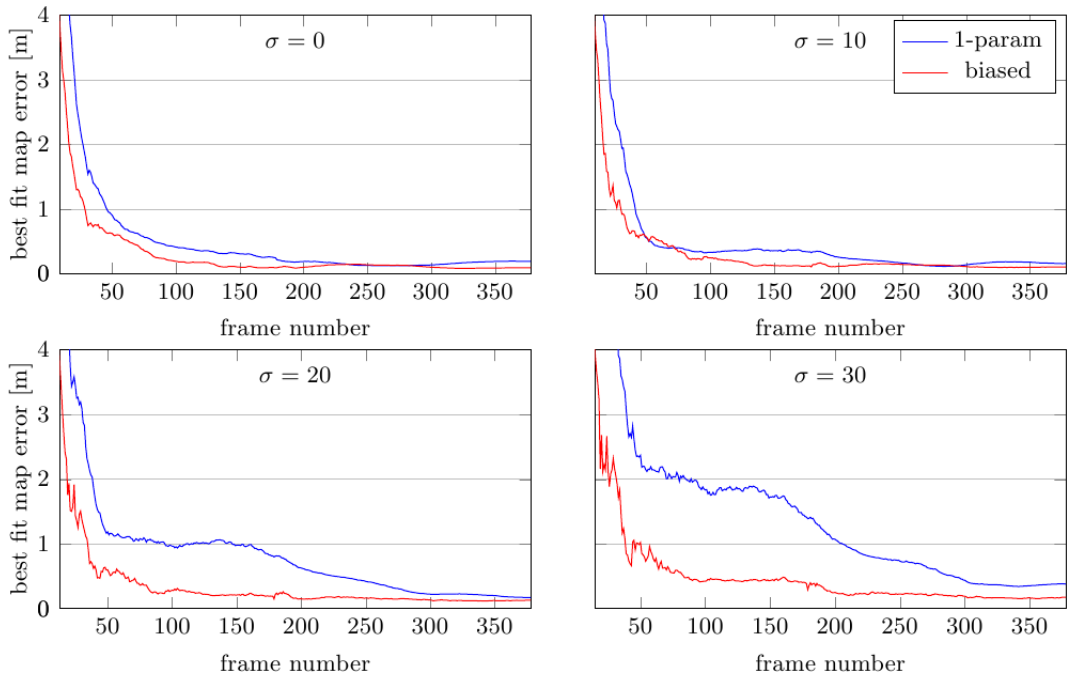


Figure 6.23.: Comparison of best-fit map error for the "stripes" sequence employing subpixel matching. Each subplot shows the result for one of the additive noise σ settings.

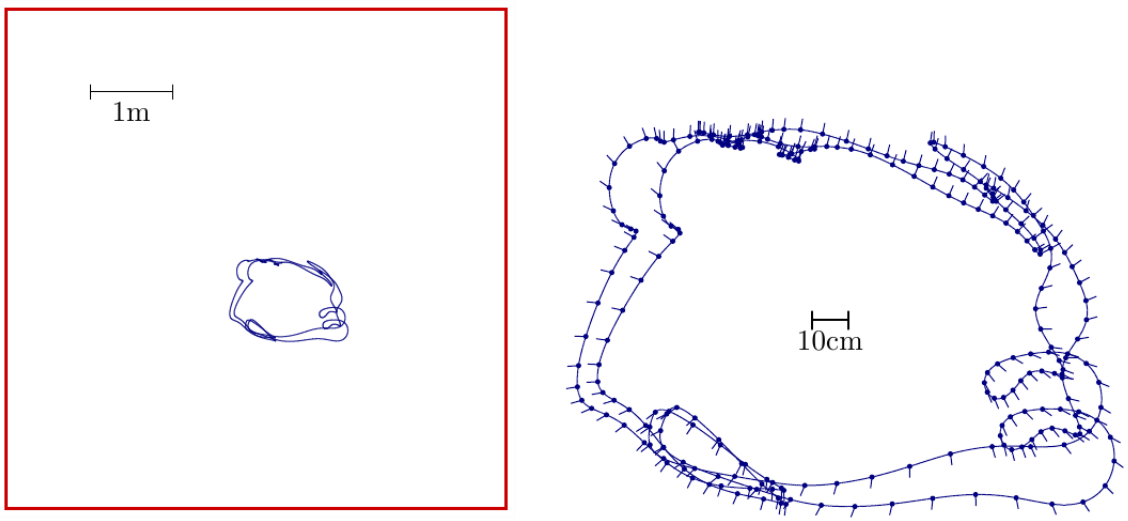


Figure 6.24.: (a) shows a schematic top view of the "box" sequence. (b) shows a close-up view of the trajectory. For every 15th frame, the viewing direction is indicated by a vector in the direction of the optical axis.

ased" model improves accuracy over the "1-param" model. However, this improvement is comparatively small. For instance, we have seen that switching between subpixel- and pixel-accurate matching has a greater influence on accuracy than switching between the models. By switching to pixel-accurate matching, we have also seen that even small deviations from the assumptions can outweigh the advantage of the "biased" model. In such cases, the rigidness of the bundle model can have a stabilising effect because it prevents adaption of the bias parameters to non-appearance bias influences.

Multiple Bias Sources

In the next experiment we move towards more realistic conditions. We use a sequence where multiple sources of bias occur, specifically position-dependent bias caused by incorrect normal estimates and bias associated with motion blur. We also use additive Gaussian noise to create appearance bias. Like before we run experiments using four different settings of additive noise as well as pixel-accurate and subpixel-accurate feature matching.

With this set of experiments we want to evaluate whether the "1-param" model stabilises the estimation through the rigid bundle structure. If so, we are interested in whether this effect counterbalances the disadvantage of not being able to adapt to appearance bias.

"Box" Sequence. Figure 6.24 shows a schematic of the "box" scene and trajectory. Figure 6.25 shows some images from the rendered sequence. The scene is a $6\,\mathrm{m} \times 6\,\mathrm{m} \times 6\,\mathrm{m}$ cube. The inside walls of the cube are texture-mapped with photographs of boxes and clutter on a shelf. The camera is placed inside the cube and is looking outwards. The

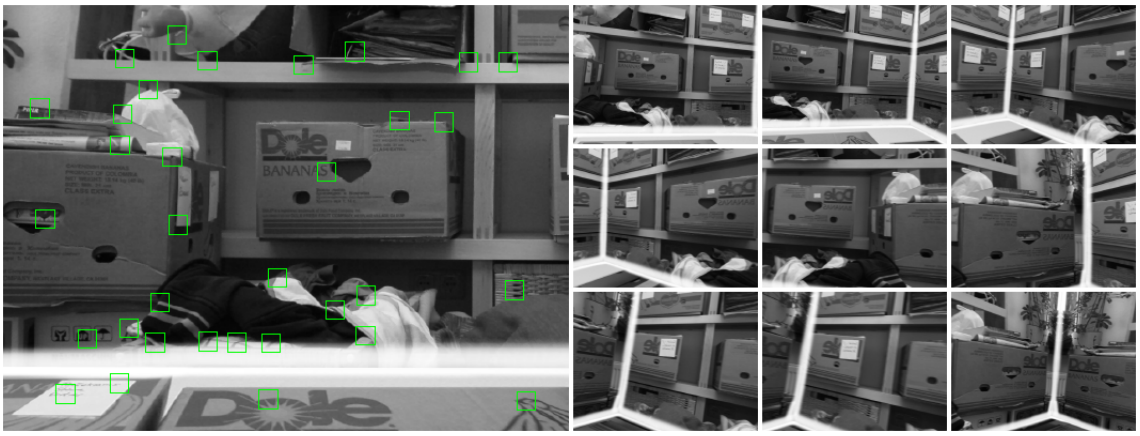


Figure 6.25.: Selected images of the "box" sequence. On the left, the first image of the sequence is shown with the selected features overlaid.

camera trajectory describes two full 360° loops. The trajectory has been reconstructed from a hand-held loop sequence, concatenated twice, and smoothed. The sequence is rendered with a camera shutter time of 33 ms. This results in mild motion blur because the camera motion is relatively slow.

This sequence is a more challenging than the previous ones. With 132 seconds ($\sim$ 4000 frames) it is considerably longer. It covers a larger environment and comprises a more realistic camera motion. No feature remains visible throughout the sequence. The SLAM system must close several loops. The first big 360° loop closure occurs around frame 1657, when features from the start of the sequence reappear.

In the experimental runs, ground truth feature normals are not provided to the SLAM system. Instead, features are initialised as perpendicularly facing the reference camera at the time of initialisation. These incorrect normal estimates will cause a systematic error in the predicted matching templates. We expect that this will result in systematic position-dependent measurement bias, which neither the "biased" nor the "1-param" model can correctly compensate. This is verified by looking at the measurement errors of individual features. Figure 6.26 shows the distribution of measurement error for three exemplary features. The plot shows the experiment without additive noise and with subpixel-accurate matching, which practically eliminates appearance bias. Clearly, we can observe systematic position-dependent bias caused by the incorrect normal estimates. The distribution of measurement errors does not have a Gaussian profile.

"Box" Sequence with Subpixel Matching. Let us first look at the results of the experiments using the pixel-accurate matching method. The absolute camera position error is smaller for the "1-param" model, cf. Figure 6.27.

The following table gives a comparison of the models with respect to the fraction of features for which the bias could be decreased. The first column gives the standard deviation of the additive noise in the sequence. The second column gives the percentage of features that have a smaller bias using the "biased" model. The third column states

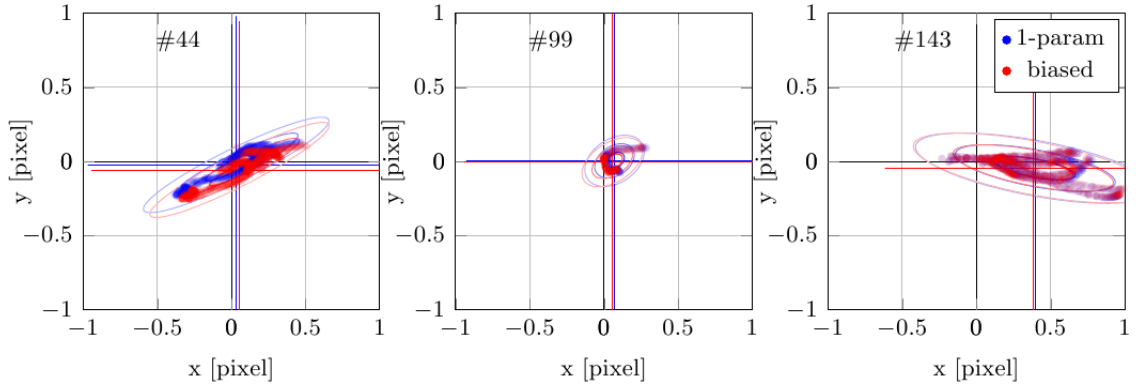


Figure 6.26.: Measurement error for selected features for the "box" sequence. The plots show the experiment without additive noise and with subpixel-accurate matching. The three plots show, for one feature each, the distribution of measurement error. The red and blue dots show individual measurements for the "1-param" and "biased" models, respectively.

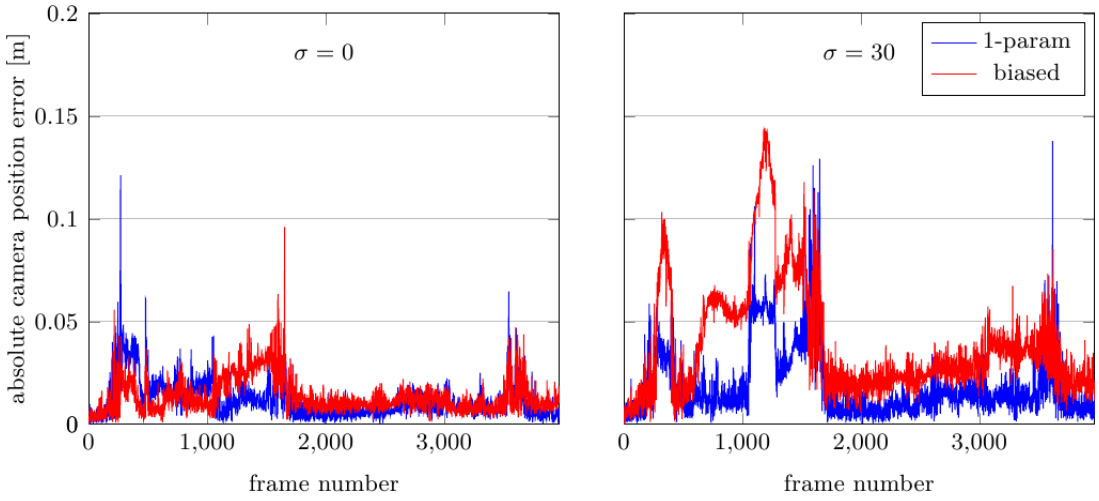


Figure 6.27.: Absolute camera position error for the "box" sequence. The left plot shows the results for the sequence without additive noise. The right plot shows the results for additive Gaussian noise with standard deviation of $\sigma = 30$ intensity levels.

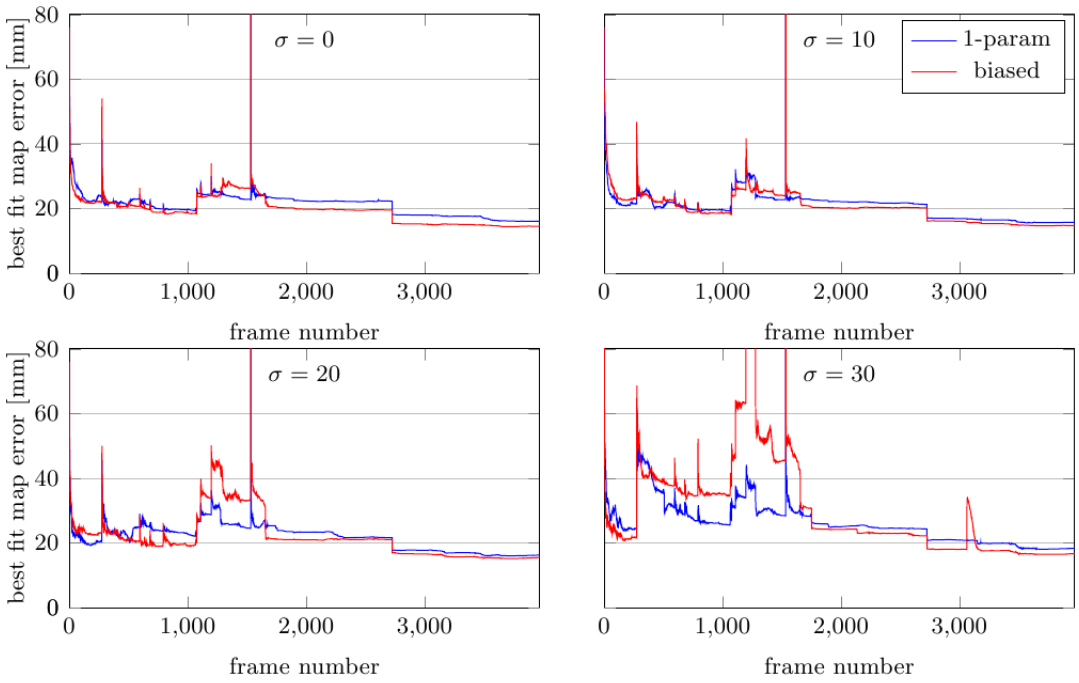


Figure 6.28.: Comparison of best-fit map error for the "box" sequence. Each subplot shows the result for one of the additive noise σ settings.

the percentage of features that have a smaller bias using the "1-param" model.

σ	"biased"	"1-param"
0	28%	72%
10	31%	69%
20	40%	60%
30	33%	66%

Here too, the "1-param" model achieves better bias values consistently. This is consistent with the previous experiments where we have seen that for pixel-accurate matching the "1-param" model achieves slightly better results.

The best-fit map error in Figure 6.28 is difficult to interpret. None of the models clearly outperforms the other. The "biased" model temporarily has larger errors but converges to smaller values towards the end of the sequence. This is different than for the previous experiments. The "biased" model seems to be partially successful in adapting to non-appearance biases.

The absolute measurement error is virtually identical for both models, cf. Figure 6.29, except for the $\sigma = 30$ case where the "1-param" model seems to have a stabilizing effect.

Let us look at the experiments with subpixel-accurate matching. Just like in the previous experiments, the errors for both models are smaller than for pixel-accurate matching. The absolute camera position error is shown in Figure 6.27. For moderate amounts of noise, the "biased" model achieves lower errors. For the $\sigma = 30$ case the "1-param" model is better.

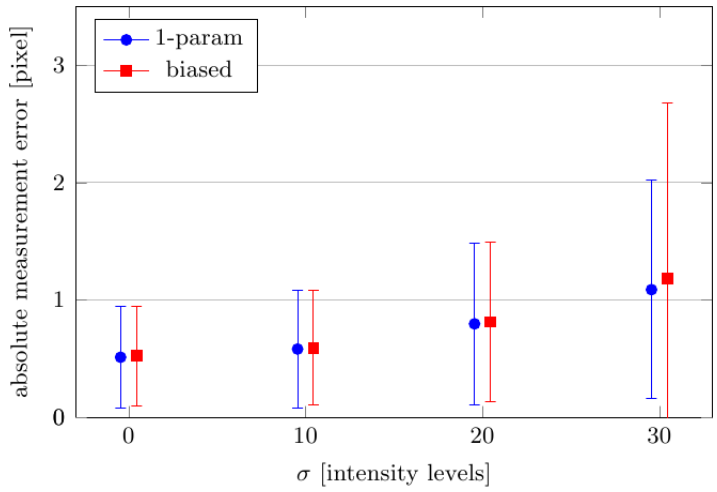


Figure 6.29.: Measurement error for the "box" sequence. Mean and standard deviation of the absolute measurement error for the "1-param" and "biased" models are compared at different additive noise levels.

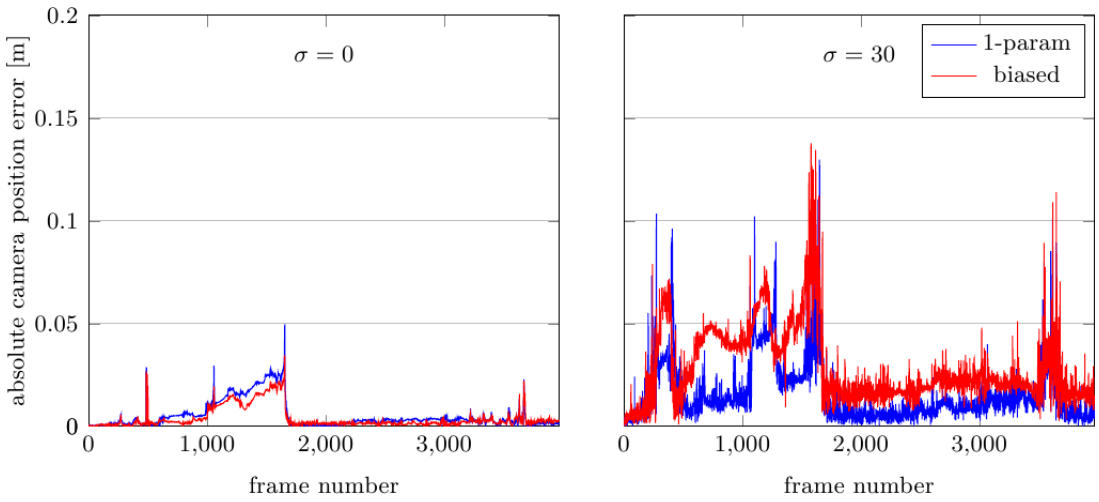


Figure 6.30.: Absolute camera position error for the "box" sequence employing subpixel matching. The left plot shows the results for the sequence without additive noise. The right plot shows the results for additive Gaussian noise with standard deviation of $\sigma = 30$ intensity levels.

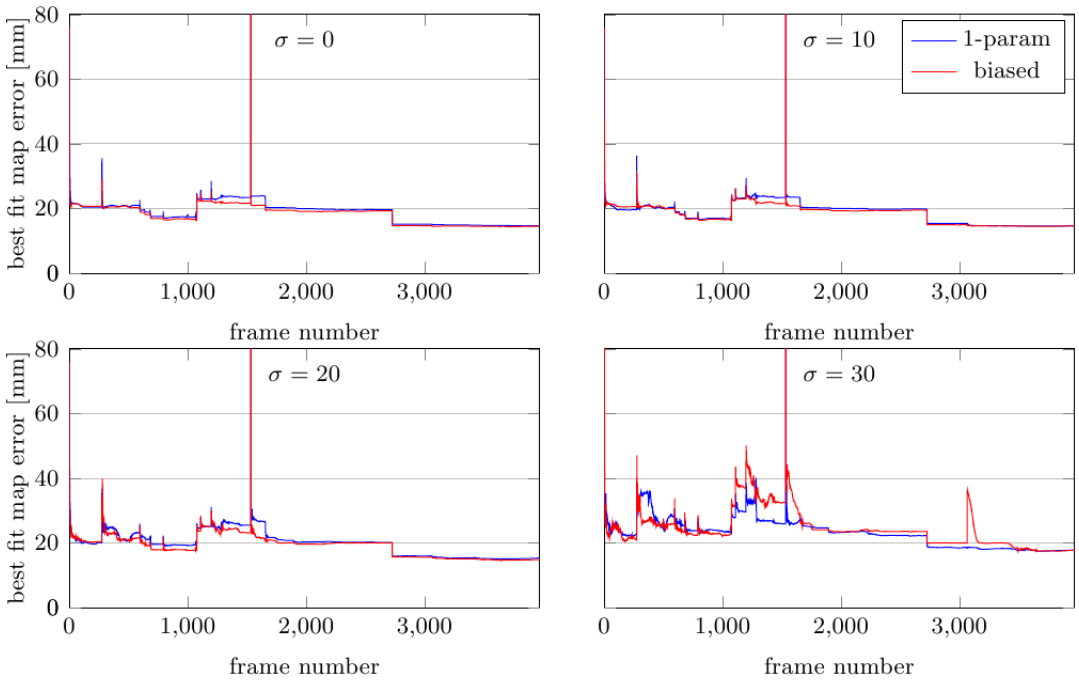


Figure 6.31.: Comparison of best-fit map error for the "box" sequence employing subpixel matching. Each subplot shows the result for one of the additive noise σ settings.

This behaviour is confirmed by the best-fit map error plots, cf. Figure 6.31. For larger amounts of noise, $\sigma = 30$, the "1-param" model is better, possibly due to a stabilising effect of the rigid bundle configuration. The same can also be observed in the absolute measurement error in Figure 6.32. For $\sigma = 30$ the "1-param" model achieves smaller errors. Otherwise, the absolute measurement error is virtually identical for both models. The comparison of the models with respect to the fraction of features for which the bias could be decreased confirms the result yet again.

σ	"biased"	"1-param"
0	65%	35%
10	65%	35%
20	58%	42%
30	43%	57%

These observations are difficult to interpret. On the one hand, the "biased" model seems to be successful in countering also non-appearance biases. On the other hand, the "1-param" model is better for large amount of additive noise. This is surprising, because increasing the noise should increase *only* the appearance bias, i.e., the bias source for which the "biased" model is designed.

A conclusion from the "box" experiment is that also for more complex scenes clearly subpixel-accurate matching gives superior results. Beyond that, we can derive no clear

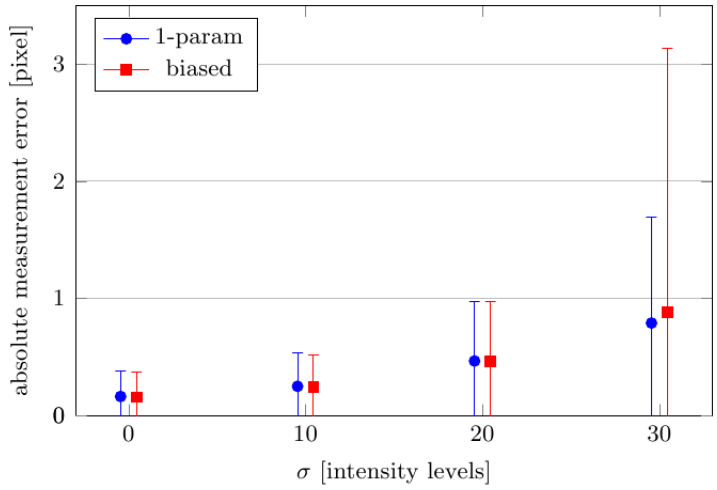


Figure 6.32.: Measurement error for the "box" sequence employing subpixel matching. Mean and standard deviation of the absolute measurement error for the "1-param" and "biased" models are compared at different noise levels.

recommendations. For moderate levels of noise, the "biased" model seems slightly more accurate than the "1-param" model. For higher noise levels the situation reverses. In general, the accuracy difference between the models is marginal.

Summary

We have presented experiments that compare the "biased" and "1-param" models. The "plane" and "stripes" sequences were designed specifically to present ideal conditions with Lambertian scene surfaces of known orientation and constant illumination. In these sequences appearance bias caused by intensity noise is the *only* violation of the bundle model assumptions. Hence, our expectation was that the "biased" model would perform better on these sequences. When sub-pixel accurate template matching was employed the "biased" model could slightly improve accuracy indeed. Interestingly, this is not the case with pixel accurate matching. Here, the "1-param" model performed slightly better. In general, the difference between the models was comparatively small. The difference between subpixel- and pixel-accurate matching is much more pronounced.

As soon as some of the assumptions are violated the advantage of the "biased" model should not be taken for granted. In the more challenging "box" experiment, the feature normals are unknown. In this case the better-performing model depended on both, the matching method and the image noise level.

We had suspected, that the added rigidness of the bundle model would provide additional stability against non-appearance bias influences, such as are present in the "box" sequence. Our experiments did not confirm this expectation as both models worked comparably well on this sequence. Similarly, Azarbayejani & Pentland (1995) suspected a reduced stability of the filter when the state vector is enlarged. We saw no indication of such problems with the "biased" model.

We can compare our results with the findings with those of McLauchlan & Murray (1995) on simulated point cloud data. They also experimentally compared 1-parameter and 3-parameter models. For their experiments, they simulated measurements by computing the ideal 2D projections of scene points and perturbing them with 2D Gaussian noise. In contrast, we use real 2D image measurements obtained by template matching while simulated noise is added directly to the rendered images. This allows to evaluate the models on measurement data that is closer to what we get from real images.

Similar to McLauchlan & Murray (1995), we found that the models are comparable in terms of camera motion estimation. We can also confirm that the best-fit map is more accurate for the "biased" model although the improvement seems not as pronounced as in their experiments. We can not confirm their observation that a 1-parameter model "rapidly reaches the point where it can no longer improve its estimates, because the remaining errors are in the fixed rays computed from the first image" (McLauchlan & Murray, 1995, p. 319). To the contrary, in the "stripes" experiment we observe a slow but steady improvement in the "1-param" map, eventually converging to the same accuracy as the "biased" model.

With regard to the experiments of Chiuso et al. (2002), we did not observe similar filter divergence, neither in the simulated experiments nor on the real sequences presented below. This confirms our suspicion, that the catastrophic failures they observed on long sequences is due to their use of only a single anchor respectively reference frame. Each new feature is transfered back to the initial anchor and in the process the error in the current camera pose estimate is transferred to the fixed feature ray. Presumably, neglecting these errors leads to the observed divergence in their work.

To summarise, we have seen that the bundle model performs similar to a model including bias parameters. Under ideal conditions, there is a minimal loss of accuracy caused by the lack of appearance-bias parameters. In realistic scenarios, this is negligible in comparison to other sources of error.

6.9.2. State Size and Processing Time

In this section, we illustrate the reduction in state size that can be achieved with the inverse depth bundle parameterisation. We use two real sequences that have been captured using a Point Grey Bumblebee® stereo camera. For processing by the SLAM system the pre-recorded sequences were read from disk. However, the processing time required to capture and rectify the images is included in the timing results given below.

The reduced per-feature state size of the bundle parameterisation can be used in two ways: We handle in real-time maps that are denser or span larger environments. The advantage in the first case is clear. We can add lots of additional features per anchor at the small cost of one parameter per feature. The usefulness in the second case depends on an initialisation heuristic that facilitates the sparse initialisation of anchors. Otherwise, if there are many anchors with only few features per anchor the advantage of bundle parameterisation is lost. In the extreme case, with only one feature per anchor, we would require more parameters than the unified inverse depth parameterisation. The following experiments illustrate that with the simple initialisation strategy proposed in Section 6.6.3,

Figure 6.33.: Selected images of the "office" sequence. On the left, the first image of the sequence is shown with the selected features overlaid.

the bundle model can achieve an effective per-feature state size of ~ 1.5, i.e., only a quarter of the space required by the unified inverse depth parameterisation.

We illustrate the state size reduction on the following two sequences. The "office" sequence is recorded in a structured office environment. The camera moves in a single room where translational motion is restricted to a volume of approximately $1 \times 1 \times 2$ meters. Figure 6.33 shows selected images from this sequence. Here, the need to initialise new feature bundles arises mainly because the camera rotates away from known features. The "office" sequence contains a 360° loop, which is reliably closed by our systen over a variety of parameter settings. Figure 6.35 shows two intermediate views of the map, before and after loop closure. The sequence is 1640 frames long (about 54 seconds). The final map contains 218 features and 21 anchors.

The "outdoor" sequence shows a more exploratory kind of motion on an open outdoor square. The camera translates forward on a path of approximately 15 meters, roughly in the viewing direction. Figure 6.34 shows a few images from this sequence. Here, the environment is less structured. Many features are of low quality, such as those on the shrubs and trees in the background or on the gravel in the foreground. Because of the forward motion, initialisation is required less often in this scenario. New feature bundles have to be initialised mainly because mapped features have moved too close to, or past the camera. The "outdoor" sequence is 765 frames long (about 26 seconds). The final map contains 132 features and 8 anchors.

Plots of the state size that is required for the map over time are shown in Figure 6.36 for the "office" sequence, and in Figure 6.37 for the "outdoor" sequence. The plots illustrate the state size that was actually required by our visual SLAM system using the bundle representation. For comparison, the state sizes that would arise for the same map using different parameterisations are shown. The plot shows the classic Euclidean parameterisation (3 parameters per feature), the unified inverse depth parameterisation (6 parameters per feature), and the bundle parameterisation with additional bias parameters (3 parameters per feature plus 6 parameters per anchor).

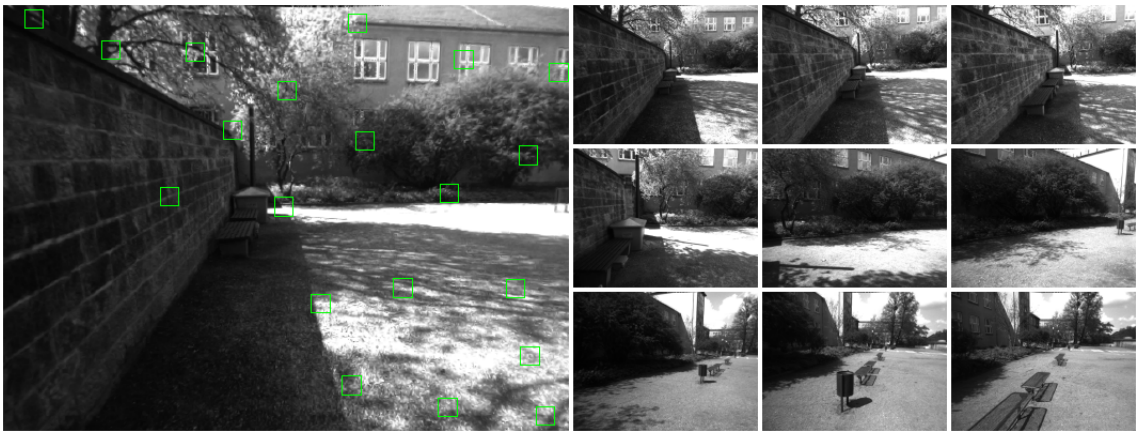


Figure 6.34.: Selected images of the "outdoor" sequence. On the left, the first image of the sequence is shown with the selected features overlaid.

Clearly, in both experiments the bundle parameterisation is effective in keeping the state vector small. Throughout both sequences the state size remains well below the hypothetical Euclidean and inverse depth state size. The effective state size per feature is 1.6 for the indoor respective 1.4 for the outdoor sequence.

With additional bias parameters, the bundle parameterisation is slightly more expensive than the Euclidean parameterisation because of the additional space required for the anchors. Still, it clearly outperforms the unified inverse depth parameterisation.

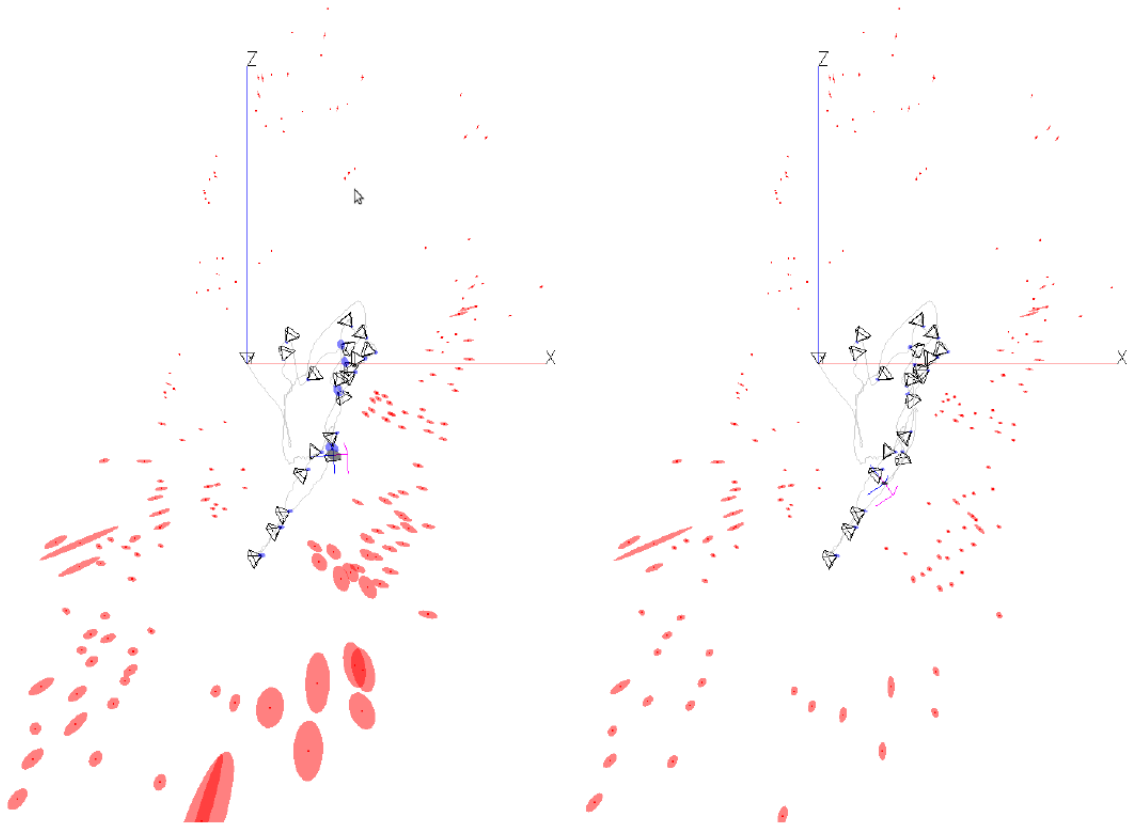
The experiments were run on a Desktop PC with a 2.8 GHz Intel® Core™2 Quad CPU. For both sequences, the processing time stays within the real-time constraint of 33 ms per frame throughout. The following table gives a breakdown of the per frame processing time for the "office" sequence. For the different processing steps the average and standard deviation is computed over the last 200 frames of the sequence, where the full map size has been reached.

image acquisition, rectification	3.5 ms	±0.1 ms
corner detection	3.5 ms	±0.2 ms
feature prediction and correlation search	2.5 ms	±0.5 ms
Joint Compatibility test	0.3 ms	±0.1 ms
filter update	6.0 ms	±1.0 ms
visualisation	4.1 ms	±0.3 ms

Occasionally, initialisation of a new feature bundle is required. Depending on the number of new features and current size of the map this takes additional $0.3 - 3$ ms.

Processing times for the "outdoor" sequence are very similar. Corner detection is a bit slower due to the unstructured environment. This is compensated by faster EKF update due to the more heavily compressed map.

To illustrate the effect of state vector size on the EKF update and overall processing time, we perform two runs of our SLAM system on the "office" sequence. In the first run, we use the bundle parameterisation to represent the map (6 parameters per anchor



(a) (b)

Figure 6.35.: Two views of the reconstructed map for the "office" sequence. Anchor poses are illustrated by small black pyramids, features are depicted by red dots. The dual blue/magenta coordinate system indicates the current camera pose. The 3σ uncertainty regions of features, anchors, and camera are depicted by filled ellipses. The uncertainty ellipses for the features include both the uncertainty of the feature with respect to the anchor and the uncertainty of the anchor with respect to the world. The views show the estimated map immediately before and after a big loop is closed. In (a) the accumulating uncertainty due to exploring new territory is visible. When re-observing old features in (b) the uncertainty is greatly reduced.

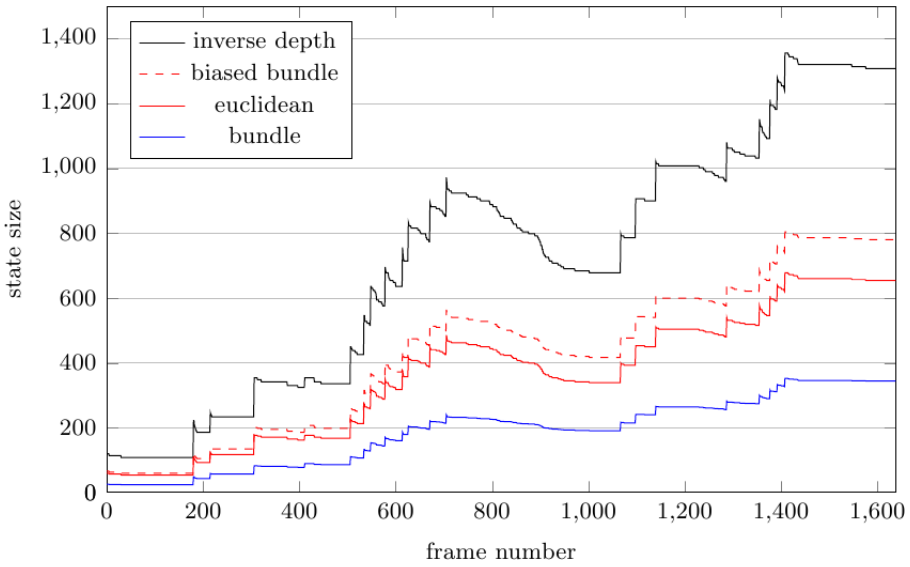


Figure 6.36.: Evolution of the state vector size for the "office" sequence for different feature parameterisations. The solid blue curve on the bottom shows the state size of the map state that is required using the bundle parameterisation. The other curves illustrate the state size that would be required for the same map when using the inverse depth, Euclidean, or biased bundle parameterisations.

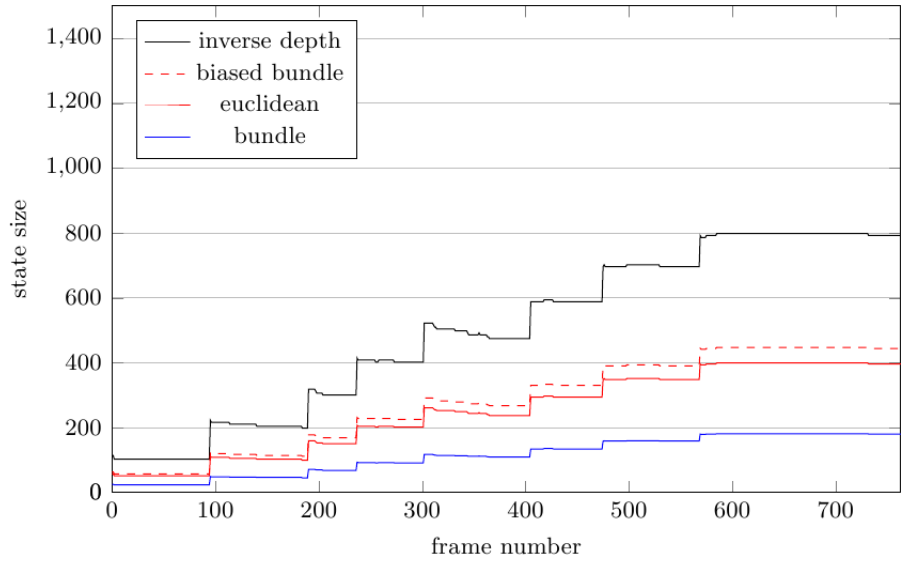


Figure 6.37.: Evolution of the state vector size for the "outdoor" sequence for different feature parameterisations. The solid blue curve on the bottom shows the state size of the map state that is required using the bundle parameterisation. The other curves illustrate the state size that would be required for the same map when using the inverse depth, Euclidean, or biased bundle parameterisations.

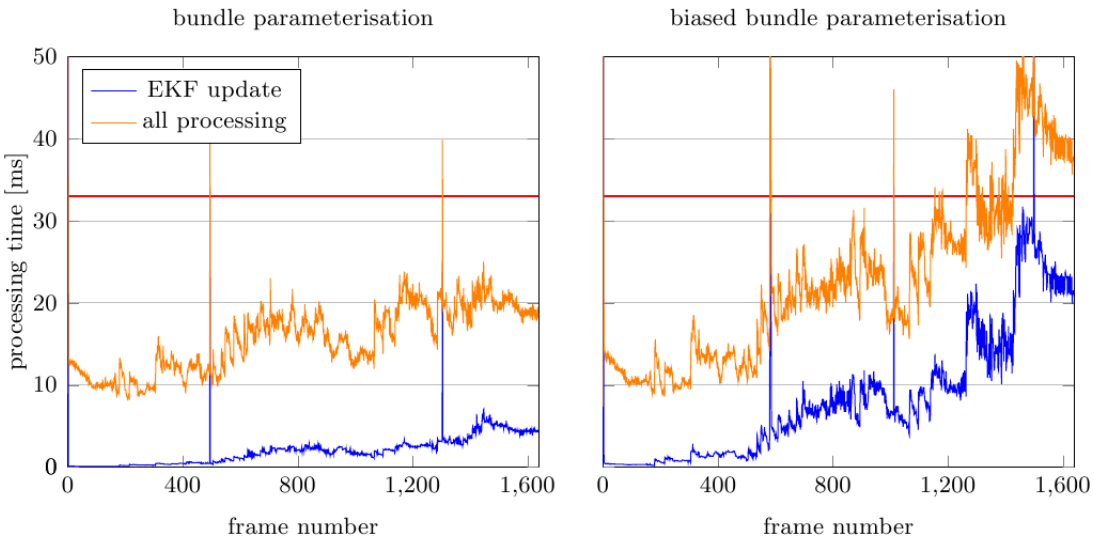


Figure 6.38.: Comparison of per frame processing times for the bundle parameterisations and the bundle parameterisation with bias parameters. The 33 ms per frame budget is indicated by the red line.

plus 1 parameter per feature). In the second run, we use the bundle parameterisation with additional bias parameters (6 parameters per anchor plus 3 parameters per feature). For both setups, Figure 6.38 plots the overall per frame processing time, as well as the time required for the EKF state update. It can be seen that while the state dimension is small, the EKF update takes up a relatively small fraction of the overall processing time. Towards the end of the sequence, the bundle parameterisation requires 344 state parameters to represent the map. The overall processing time remains well below the 33 ms real-time limit, and the EKF update makes up for at most a quarter of this.

With additional bias parameters, the same map requires 780 parameters. On the right-hand side of Figure 6.38 we can see that in this case the sequence can no longer be processed in real-time. The EKF update alone consumes the time budget almost completely.

Assuming that image processing and visualisation take up roughly the time given in the above table, we can sustain real-time operation for a map state size of ca. 450. This translates to 75 inverse depth features, 150 Euclidean features, or 300 bundle features. The bundle parameterisation allows maps that are 4 times as big (or dense) as the inverse depth parameterisation.

6.10. Discussion

In this chapter, we have explored a representation that significantly increases the number of point features that can be handled in real-time EKF visual SLAM. We have proposed the inverse depth bundle parameterisation, that groups features by their common frame of initial observation. This allows parameter sharing between features which reduces the size of individual features in the state vector.

Having more features allows to construct maps that span larger environments or maps that are denser. The bundle representation is suited to both tasks. Employing it to obtain denser maps is straightforward, as adding more features for each anchor is simple and cheap. Building maps that span larger environments requires more consideration. A map management strategy is required that keeps the number of features per anchor high while at the same time maintaining a sparse map. We have proposed a simple feature initialisation heuristic that achieves this goal.

We have presented experiments on real image data to illustrate the benefits of our parameterisation. Of the above mentioned scenarios, we considered the more challenging case of building a sparse map. In the experiments we achieved an average state size of 1.5 parameters per feature. With our implementation, this allows maps of about 300 features to be handled in real-time on current desktop hardware. This translates to a 4 times increase in map size over the unified inverse depth parameterisation. If the goal is to construct denser point maps, we will add as many features as possible per anchor and thus the state size of a feature would be further reduced.

Our bundle model achieves its efficiency by representing each feature by only one parameter relative to its anchor. This removes two degrees of freedom from the 3D position of the feature point. These degrees of freedom are not eliminated from the state, however. Instead they are moved to the anchor and thus shared among several features. This raises the question whether this is an adequate representation or if it implies losses in mapping accuracy. To this end, we have presented a detailed analysis of the feature measurement process via template matching. This process can be viewed as the back-projection of a feature template onto the scene surface and subsequent projection into the current camera image. We emphasise the special role of the initial observation of a feature. The initial observation establishes the feature template that is used for template matching subsequently. Thus, it is appealing to view the initial observation not as a measurement in itself but merely as establishing the basis for future measurement. To our knowledge, this is an insight that has been overlooked in visual SLAM to date. Moreover, we expound on the closely related issue of measurement bias, which has not been investigated in visual SLAM before, either.

Our SLAMDUNK framework allows to substantiate this discussion with experimental analysis. In particular we augmented the bundle model with two additional degrees of freedom for every feature and compared this "biased" model with the original one. In theory, the intensity noise in the template image is the one source of measurement bias that is addressed by the "biased" model but not the bundle model. All other effects are handled or ignored by both models uniformly.

In our experiments we observe that the expected advantage of the "biased" model is marginal even under idealised conditions, i.e., intensity noise is the *only* source of bias, and with highly accurate subpixel measurements. Switching to pixel accurate matching or more realistic conditions may even reverse the situation and make the bundle model perform minimally better. Our general observation is that the matching method and the accuracy of the predicted template have a much more significant influence than the choice of model. To give an example, we consider the average measurement error for images with moderate noise level of $\sigma = 20$. For the "box" sequence with pixel-accurate matching and unknown feature normals, the average measurement error is 0.8 pixels for the

bundle model and 0.81 pixels for the biased model, respectively. The difference between the models is 1/100th of a pixel. In contrast to that, improving appearance prediction through accurately known normal vectors such as in the "plane" sequence reduces the average measurement error to 0.4 pixels. Additionally employing subpixel matching gives a further reduction to 0.1 pixels. We can conclude that available processing time is much better spent on estimating surface orientation or subpixel refinement than on the additional EKF update time required for the bias parameters.

In this chapter, special emphasis has been put on the accurate prediction of feature appearance, i.e., on the prediction of accurate matching templates for the current image. For this purpose it has been useful to explicitly represent the reference camera pose (the anchor) in the feature state. The same idea will also be crucial for the planar feature model discussed in the next chapter.

We have also seen that a good estimate of the surface normal of a feature improves appearance prediction and thus matching accuracy. The estimation of normal vectors would be a desirable extension to our current system. An estimate could be obtained for example by analysing the disparity field in the initial stereo pair. Alternatively, the normal vector can be included into the probabilistic feature state and sequentially updated in the EKF. This approach will be pursued in the next chapter as well.

7 Planar Features For Visual SLAM

7.1. Introduction

An appealing approach to increase the density of visual SLAM maps is to move from points to higher-level, more descriptive features. In this chapter, we explore the use of planar segments of the environment as features in visual SLAM. This requires more profound changes to the feature and measurement models than merely increasing the density of point features, as discussed in the previous chapter. While increasing point density is a straightforward approach to increase map density, there are fundamental limitations which will be discussed in the following. The motivation for using higher-level features, and planes in particular, is threefold.

First, point-based representations are not well suited to certain tasks. Consider for example the rendering of occlusions in augmented reality. For credible augmentation, a virtual object that is inserted into the real scene, should be occluded by real objects that are located in front of it. A point map of the real scene, even a dense one, can not be directly used to compute such occlusions. The point map will have to be converted first to a representation that is appropriate for this task, for instance a polygonal mesh of the environment. Another example is robotic path planning, which requires reasoning about free-space. In this case, a volumetric representation would fit the task. Point based representations are not adequate when we want to predict, simulate, or understand physical interactions and geometrical relationships in the scene.

Second, representing the scene as points is not *efficient* in terms of state size. By using more descriptive features, such as line or surface segments, larger parts of the scene can be represented in a compact, less redundant form. In this chapter we have chosen surfaces over line segments because they seem to be better suited to the above-mentioned tasks than line segments. However, we need to make additional assumptions to allow for tractable representation and update of the environment model. In particular, we assume that surfaces are piecewise planar. Especially in man-made environments, planar

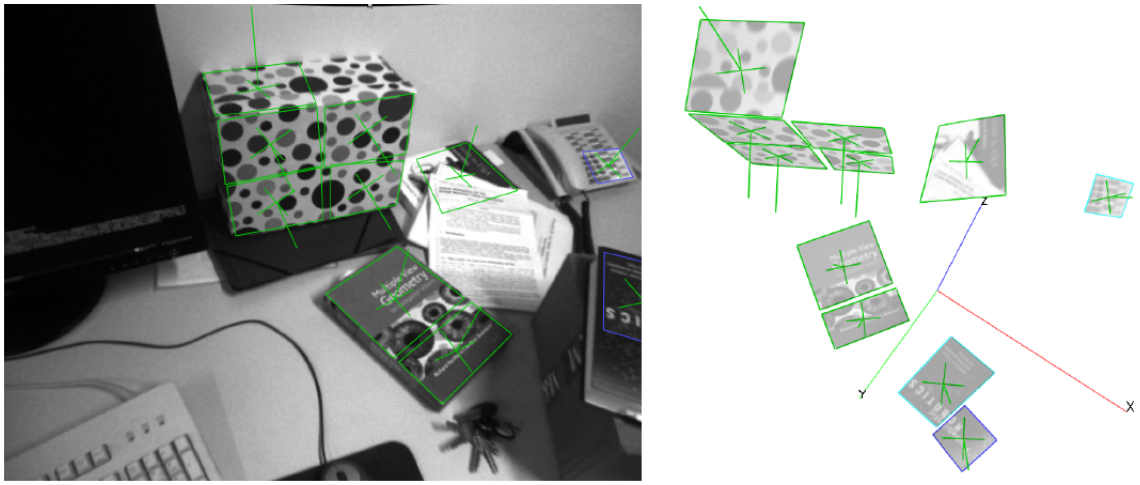


Figure 7.1.: An example of SLAM with the planar features proposed in this chapter. On the left, a frame of a desktop video sequence is shown. The mapped planar features are overlaid on the image, with an indication of their outline and normal vector. On the right, the map is shown from a different viewpoint.

structures are common and may allow a compact representation of large parts of the environment.

Finally, the camera images contain more information than is exploited by point matching. Planar features allow us to use more of this information by measuring the homography induced by the feature plane in the image. For point features that lie on approximately planar scene surfaces, homography-based appearance prediction greatly improves the stability of the template matching procedure by removing the effects of varying viewpoint. This is especially the case if an accurate estimate of a feature's normal vector is available, which has been clearly demonstrated in the experiments of Chapters 5 and 6. Moreover, and this is not utilised by point matching, changes in feature appearance also provide information which can be directly used to improve the state estimate. For instance, if we observe that the scale of the observed image patch is larger than we expected, this tells us that the camera is closer to the feature than we predicted. Also, the amount of perspective distortion is directly related to the relative orientation between the camera and scene surface. To exploit such information, modeling a feature measurement as a single 2D point coordinate is no longer sufficient.

In this chapter, we propose a probabilistic map representation for planar surface segments. Similar to the view-point based feature model, such a segment is described in terms of its appearance in the image where it was initially observed, and the anchor pose of the camera when this initial image was taken. We propose a method to measure these planar features using directly the image intensities of individual pixels in the camera images. In this way, the information provided by changes in feature appearance due to changing view-point is immediately used to improve the state estimate.

This direct method greatly increases the measurement accuracy with respect to template matching. However, it also poses several new challenges. Linearising pixel intensity

measurements, as required by the EKF, is problematic because image functions are often highly nonlinear. This severely restricts the amount of image motion that can be handled by the straightforward application of the EKF update. This has been a major point of criticism with previous work. To deal with this issue, we propose an iterative measurement update step to obtain the same robustness to image motion as for point feature matching. The update is initialised using standard template matching from the view-point based feature model. The result is iteratively refined using direct intensity measurements.

A second problem is that the measurement vector is potentially very large. The dimension of the measurement depends on the size of the image area where a planar feature is visible. Measurements commonly comprise thousands of pixel intensities. This leads to serious performance problems if these measurements are used in a straightforward way because the EKF update has cubic complexity in the size of the measurement vector. We present a method of reducing high-dimensional measurements to *fused measurements* of bounded size. The cost of this reduction is linear in the measurement size. Thus the complexity of the update step is reduced from cubic to linear in measurement size. This novel reduction operation is derived using the duality of the Kalman Filter and the Information Filter. While we present measurement fusion in the context of planar feature measurements here, this result is of broader interest because the methodology is generally applicable in situations where measurement vectors consist of many conditionally independent parts. In SLAM and tracking applications, we expect this to occur frequently with higher-level features such as lines, planes, or object models. Measurement fusion offers a principled, generally applicable way of handling these situations in the EKF framework.

The performance of visual SLAM using planar features is evaluated using simulated and real image sequences. Using simulation experiments we show that planar features provide increased accuracy in comparison to traditional point features, even when only a single planar feature is used. We use real image sequences to demonstrate robust operation in indoor environments. We further show that dense maps of piecewise planar scenes can be constructed, which are useful for instance for augmented reality applications.

This chapter is structured as follows. In Section 7.2 we review related work. After illustrating the use of image intensities as direct measurements of the state in Section 7.3, we go on to introduce the representation and measurement method for planar features in Section 7.4. The measurement fusion method is derived in Section 7.5. The iterative measurement process is detailed in Section 7.6. We describe the initialisation of planar features in Section 7.7. Further implementation details are discussed Section 7.8. Section 7.9 presents an experimental evaluation of planar features. We conclude with a discussion in Section 7.10.

7.2. Related Work

7.2.1. Planar Features

Existing approaches to incorporate planar segments as visual SLAM features can be split into two groups according to the measurement model used. First, a planar segment can represent a group of point features. The measurement of the plane is then provided by extracting and measuring those point features. Second, we have methods that directly

represent planar segments by their image texture. The measurement of the plane uses pixel intensities in the image.

In the first group, Gee et al. (2007) present a visual SLAM system in which planar structural components are detected and embedded within the map. After building a map of point features, subsets of these points lying on common planes are identified. Subsets of points can then be more compactly represented by extracting common parameters in a plane representation. These planes are not directly observable in the camera images. Instead, they are inferred from classical point measurements. The method relies solely on sparse information from existing point features. Thus, detected planes may not correspond to physical scene planes.

Martínez-Carranza & Calway (2009b) also use planar features in an EKF based visual SLAM framework. Their approach is closely related to ours. They adopt the planar feature parameterisation proposed in this thesis, however, they use a different measurement process: Update is based on matching point features on the planes. These features are adaptively selected according to the current camera pose and to the predicted visibility fields for the associated scene patches. Importantly, these matched point features are not represented in the state vector. They are defined with respect to key frames and projected into the current frame via the planar feature. Thus, the effective density of the map is increased while only needing to represent the single plane in the state vector. We have not performed a direct comparison of our measurement method and that of Martínez-Carranza & Calway (2009b). On the one hand, our approach of using directly pixel intensities as measurements potentially provides higher accuracy, which is indicated by the experiments in Section 7.9.2. On the other hand, their measurement method provides straightforward handling of partial occlusion, which is not yet handled by our approach at all.

Our measurement method falls within the second group mentioned above: We use pixel measurements directly to update estimates of plane parameters. This second category is often referred to as *direct methods* in the computer vision literature. The overview by Irani & Anandan (2000) defines direct methods as "methods for motion and/or shape estimation, which recover the unknown parameters directly from measurable image quantities at each pixel in the image. This is in contrast to the feature-based methods, which first extract a sparse set of distinct features from each image separately, and then analyze their correspondences in order to determine the motion and shape" (Irani & Anandan, 2000, p. 267).

Among direct methods, the approach of Jin et al. (2003) is the closest related to our method. They use planar features in an EKF-based Structure from Motion framework with a direct measurement method. However, by linearising the measurement model, they rely on the linearity of the image gradient which limits their approach with respect to image motion and tolerable camera acceleration. We solve this issue by casting the measurement update in an iterative framework.

Planar features in (Jin et al., 2003) are represented with respect to a global coordinate frame. In contrast to our approach, the reference camera pose corresponding to the feature template is not explicitly represented. Instead, the feature template is back-projected to the global coordinate frame upon feature initialisation. This corresponds to fixing the reference pose estimate when the feature is initialised. This introduces a systematic error because the reference pose estimate cannot be updated later on. In their Structure from Motion framework where features are forgotten as soon as they move out of the image

this is not too harmful. It merely adds to accumulated drift in the estimated trajectory. In a full SLAM system however, it is potentially disastrous because it precludes correct propagation of loop closure constraints to the map.

Furthermore, Jin et al. (2003) do not handle the problem of large measurement vectors. Consequently the method is not reported of running in real-time. Finally, our feature model employs an inverse-depth parameterisation which has been shown to be more suitable for linearisation.

Silveira et al. (2008) formulate visual SLAM as a nonlinear image alignment task. They model the environment as a collection of planar surfaces and obtain the camera pose, scene structure, and illumination changes directly using image intensities as observations. Efficient second-order minimisation (Malis, 2004) is used to solve the optimisation problem. Using an iterative second-order method enlarges the region of convergence and greatly improves on the restriction to very small inter-frame displacements of Jin et al. (2003).

They parameterise planar patches by the inverse depths of three points. This seems an interesting alternative to the parameterisation proposed in this thesis. New planes can be added to the map throughout the sequence. Similar to our approach, these planes are represented with respect to reference frames. The paper is unclear as to whether the relative pose of the reference frames is part of the subsequent estimation steps or whether it is fixed. Outlier rejection is accomplished by thresholding on photometric and geometric dissimilarity measures.

A drawback of their method is the lack of a fully correlated map. Although an EKF is used for prediction, Silveira et al. (2008) is not a filtering approach. Correlations between features are not maintained which implies that loop closure constraints cannot be propagated to the map. Moreover, prior information is not exploited in the minimisation. Past information is only used to provide an initial guess for the parameters in the current image. This is in contrast to our approach, where filtering and image alignment are tightly integrated: Measurements are propagated through the fully correlated map, and prior information is balanced against image measurements in a fully probabilistic approach.

Finally, related work includes methods that employ plane-estimation techniques to improve appearance prediction for traditional point features. Molton et al. (2004a) regard feature points as small locally planar patches and estimate their normal vectors. However, this is done outside of the SLAM filter, thus ignoring correlations between normal vectors and rest of the state. While being useful for improving the observability of point features, their approach does not scale to larger planar structures because these correlations cannot be neglected in this case.

Similarly, Wuest et al. (2008) sequentially estimate the normal vector of a planar patch feature assuming known camera motion. The underlying tracking/mapping system is a different one, decoupling mapping and tracking, and lacking a fully correlated map. In addition to normal estimation, the reference template image of the feature is refined over time. Furthermore, a mask image is created to identify template pixels that do not belong to the planar region. This is useful, e.g., if the template region contains a depth discontinuity.

7.2.2. Planar Tracking and Image Alignment

Direct measurement methods often can be viewed as solving *image alignment* problems. The goal is to find the unknown parameters of a known warp function that best bring in alignment two image regions. The warp function maps positions in a template image to positions in a destination image.

The first gradient-based image alignment method was proposed by Lucas & Kanade (1981). They solve the problem by linearising the sum of squared differences between the images to iteratively improve the parameters of the warp function. The linearisation is performed in the destination image to obtain an additive increment to the parameter vector. In the terminology of Baker & Matthews (2004) this is classified as a *forwards additive* method.

Baker et al. (2001) formulate a more efficient *inverse compositional* solution. Here, the linearisation is performed in the template image and an compositional warp update is obtained. This has the advantage, that the linearisation and expensive parts of the solution must be computed only once.

Molton et al. (2004b) formulate a probabilistic extension of the inverse compositional method. They propose a solution that can be viewed as an approximation to the Extended Information Filter (EIF). This is tightly related to our measurement fusion approach in Section 7.5, which exploits the duality between the EIF and the EKF. Our solution can be classified as an *inverse additive* method.

(Malis, 2004) propose an efficient second-order approach to image alignment. This is also an iterative solution. In contrast to the first-order methods discussed above, they use a quadratic approximation to the sum of squared differences error function. This formulation provides both, higher convergence rate and convergence frequency. In contrast, our measurement method is first-order, which is a consequence of using the EKF. While second-order methods are attractive for their convergence properties, it is unclear whether and how they can be integrated within EKF based estimation.

Several authors have parameterised the image alignment warp using known scene planes and/or camera pose. Hager & Belhumeur (1998) present planar tracking using affine parameterisation as an application of their efficient forward additive method. Buenaposada & Baumela (2002a) extend this work to a fully projective homography parameterisation.

Cobzas & Sturm (2005); Cobzas et al. (2009) use inverse compositional alignment to track a camera in a scene consisting of multiple known planes. They parameterise the warp function by the camera pose. Their system also comprises a bootstrapping phase, where the parameters of user-selected planes are estimated from 2D tracked point features. In the second phase, the recovered planes are used for tracking.

Benhimane & Malis (2004) use second-order minimisation to perform planar homography tracking. Mei et al. (2008, 2006) extend the method to omnidirectional cameras and multiple planes. They parameterise the warp function by the camera pose and the plane parameters.

None of the above approaches can be classified as a full SLAM solution, because all of them are non-probabilistic and lack a correlated representation of camera and scene.

7.2.3. Plane Detection

The approach presented in this chapter does not propose a particular method of *detecting* planar regions. In our experiments, the planar features were initialised manually, by selecting their outlines in a reference image. However, automatic detection of planar regions would of course be an important part of a complete visual SLAM system. Several approaches for plane detection have been proposed in the literature. The following are representative recent examples, any of which could be used in conjunction with our system.

First, there are non-sequential methods that extract planes from a given set of two or more images.

Silveira et al. (2006) use a progressive voting procedure to identify planar regions in two views. Votes are based on triples of point feature correspondences between the views. After a plane candidate has accumulated a large enough score, a new planar region is defined by the convex hull of all point matches that agree with this hypothesis.

Fraundorfer et al. (2006) recover planar scene regions from two or more images. Seed regions are detected by affine-invariant matching of sparse features. Then, region growing is employed to expand and refine these regions. This region growing is performed using a similarity on pixel neighbourhoods to obtain pixel accurate region outlines.

Second, there are sequential methods which accumulate information over many frames. Conceptually, these methods seem a better fit to the SLAM scenario, and indeed they are often constructed on top of point-based SLAM systems.

An excellent recent example is the work by Martínez-Carranza & Calway (2009a). They present an hypothesis testing approach for detecting planar structure sequentially during real-time visual SLAM. A 2D Delaunay triangulation of mapped point-features in a reference frame provides plane hypothesis. Salient points in the hypothesised planar segment are matched in successive frames. The matching error with respect to the predicted points under the planarity assumption is analysed with a test statistic to verify or reject the plane hypothesis. Notably, the test is adaptive, i.e., based on the covariance information from the SLAM filter, providing consistent and robust planar structure detection.

7.2.4. High-Dimensional Measurements

In Section 7.5 we develop an approach to efficiently handle high-dimensional measurement vectors such as arise from planar feature measurements. A further characteristic of these measurements is that the components of the measurement vector are conditionally independent.

Straightforward application of the measurement in the Kalman Filter update requires $\mathcal{O}(m^3 + mn^2)$ operations, where m is the dimension of measurement vector and n is the dimension of the state vector. With growing measurement size, the cubic cost in m quickly renders the straightforward update infeasible.

Several approaches to tackle the problem exist in the literature. First, it is possible to apply the measurement components one at a time as scalar measurements (Brookner, 1998, p. 386). This avoids the inversion of the innovation covariance matrix which causes the cubic dependency on m. The resulting algorithm is $\mathcal{O}(mn^2)$, requiring m updates of the full state vector which is still too expensive for our application.

In contrast, our approach requires $\mathcal{O}(ma^2 + an^2)$ operations, where a is the dimension of the sub-state on which the measurement depends directly. That is, a is bounded by a constant, and $a \ll n$.

A combination of scalar measurements with postponement techniques (Knight et al., 2001; Guivant & Nebot, 2001) might be used to restrict the scalar updates to the a-dimensional sub-state and postpone the full state update until all components have been processed. To our knowledge, this combination has not been explored in the literature so far. The approach would achieve the same complexity as our method. However, it still would require m scalar updates of the sub-state, which would be slower than measurement fusion by a constant factor of 3.[1]

It is well known that the measurement update step in the information form is computationally simple (Bierman, 1977, p. 27), in our case $\mathcal{O}(ma^2)$. Our construction of fused measurements employs the information update formulation. However, recovering the state mean from the information form is $\mathcal{O}(n^3)$ in general. Thus, simply exchanging the Extended Kalman Filter for an Extended Information Filter would increase the quadratic dependency of SLAM on the map size to a cubic dependency. Efficient approximate information form SLAM algorithms exist (e.g., Thrun et al., 2004). However, we will not explore this branch of work here because the focus of our work is to make the measurement update efficient in the covariance form.

Finally, Bayard & Brugarolas (2004, 2005) propose a method to compress measurements using the QR factorisation of the stacked Jacobian. Their method was independently developed. It is of the same complexity as our method and constitutes an interesting alternative. The two approaches are closely related and we will make this connection explicit in Section 7.5.4. Both approaches have their respective advantages and disadvantages. On the one hand, the QR factorisation is numerically better conditioned and thus the method of Bayard & Brugarolas (2004) may provide higher numerical accuracy. On the other hand, our method is faster by a constant factor and straightforward to parallelize.

7.3. Direct Intensity Measurement Models

Before we introduce the planar feature model, we want to illustrate how direct pixel intensity measurements fit with the EKF framework. We will look at two features of the EKF update: First, measurements are described by a generative model. Second, the innovation, i.e., the difference between the actual and the predicted measurement, is used to update the state. With respect to these two points, we will review point measurements and extend the ideas to the direct intensity measurements used for planar features.

The generative measurement model describes measurements as a function of the state vector. That is, the model explains how a measurement is "generated" from the state. Importantly, the measurement can be *predicted* by applying the model to the current state estimate.

[1] By counting floating point operations of the involved matrix and vector operations, we find, that the sequential update of the sub-state of dimension a with m scalar measurements requires on the order of $3a^2m$ operations whereas measurement fusion only requires a^2m.

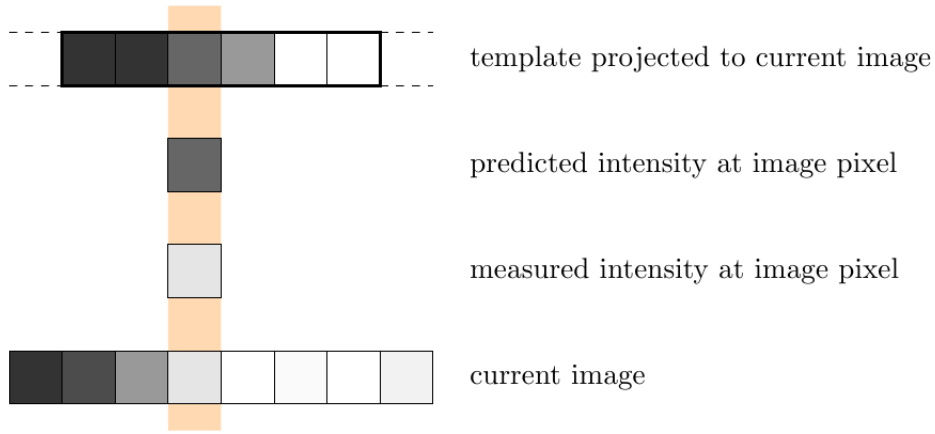


Figure 7.2.: Illustration of the update using intensity measurements. For the pixel in the highlighted column, the measured intensity is lighter than the predicted intensity. This measurement will "pull" the template to the left such that the template intensity which is projected onto the highlighted pixel then will become lighter and better match the measured intensity.

For a point feature, this means predicting the 2D position of the feature in the image. This is done by projecting the estimated state mean through the measurement function. The 2D position measurement is extracted from the current image using bottom-up image processing. The measured position is the image location that best matches the feature template.

For a planar feature, we will use the image intensities as measurements. For every pixel, a 1D measurement model describes the intensity value at that pixel as a function of the state. Similar to point features the measurement can be predicted. In the case of a planar feature, we predict the intensities of all pixels in a region of interest, namely the image region where we expect the projection of the feature. For every pixel, this can be done by predicting which template pixel comes to lie on this image pixel and looking up the corresponding intensity value.

The EKF uses the innovation, i.e., the difference between the prediction and measurement, to update the state estimate. For point features, this is the difference between the predicted and measured 2D image projection of a feature. Informally speaking, in the update, the state is pulled in a direction that moves the projected feature position closer to the measured position. States in which the projected feature position is closer to the measured position are made more likely by the measurement.

For planar features, the innovation at a specific pixel is the difference between the predicted and measured intensity. In the update, the state is pulled in a direction that moves the predicted intensity closer to the measured intensity.

This is illustrated in Figure 7.2 using a simplified one-dimensional example. The bottom row shows the current image, i.e., the measured intensities at all pixels. The top row shows the predicted image. This predicted image is obtained by placing the template, in this case a pattern of 6 pixels, at the predicted position. Corresponding pixels in the predicted

and measured image are shown in the same vertical positions. That is, if the prediction would perfectly match the current state (and if there was no noise in the image) intensities in the top and bottom row would match perfectly.

Consider the pixel location in the highlighted column. The predicted intensity at this location is a dark grey, as shown in the second row. This is the intensity of the template pixel that is (predicted to be) projected at this location. The measured intensity at this pixel is a light grey, as shown in the third row. In the EKF update, the state is pulled in a direction such that the predicted intensity moves closer to the measured intensity, i.e., such that the pixel becomes lighter. The template gradient at the highlighted pixel is positive: pixels get lighter to the right. The update makes state more likely where the template projection appears further to the left, i.e., the template is pulled in the opposite gradient direction. This is because in these states a lighter template pixel will appear at the highlighted pixel location. Thus, the prediction will better match the current image.

What makes direct intensity measurements appealing is the rigorous top-down approach. The high-level state estimate is directly connected to the low-level raw image information in a fully probabilistic manner. Moreover, more of the image information is utilised. For a point, the measurement has two dimensions. Here, the dimension of the measurement is the number of pixels covered by the template. On the one hand, this leads to more accurate updates. On the other hand, of course, it makes the update more expensive.

7.4. Representing and Measuring Planar Features

7.4.1. Planar Feature Parameterisation

A planar feature represents a plane segment in the scene. Our parameterisation of a planar feature is almost identical to the view-point based parameterisation introduced in Section 6.5. The only modification concerns the normal vector of the feature plane. For the viewpoint-based model, the normal was fixed, respectively assumed known. For planar features, the normal vector estimate will be continually improved by image measurements. Thus, the normal vector is made a part of the state vector representation.

A planar feature consists of a feature state $\mathbf{y}$, a feature ray $\mathbf{m}^{\mathcal{A}}$, a template image T, and a feature outline L. In the probabilistic state, the feature is represented by the 9-dimensional vector

$$\mathbf{y} = \begin{pmatrix} \mathbf{c} \\ \boldsymbol{\phi} \\ \rho \\ \boldsymbol{\theta} \end{pmatrix}. \tag{7.1}$$

The parameters $\boldsymbol{\phi}$ and $\mathbf{c}$ define anchor frame $\mathcal{A}$, i.e., the camera pose at the initial observation of the feature. The 3-vector $\mathbf{c}$ is the position, and the 3-vector $\boldsymbol{\phi}$ is the rotation in exponential coordinates of this initial camera pose. The parameter ρ is the inverse depth of the feature plane along the ray $\mathbf{m}^{\mathcal{A}}$. Finally, $\boldsymbol{\theta} = (\theta_x, \theta_y)^\top$ is the normal vector of the feature plane, encoded in polar coordinates.

The planar feature model is illustrated in Figure 7.3. In the scene, the feature is a segment of the (infinite) feature plane described above. The appearance and shape of the feature is described by T and L. The template image $T\colon \mathbb{R}^2 \to \mathbb{R}$ stores the reference

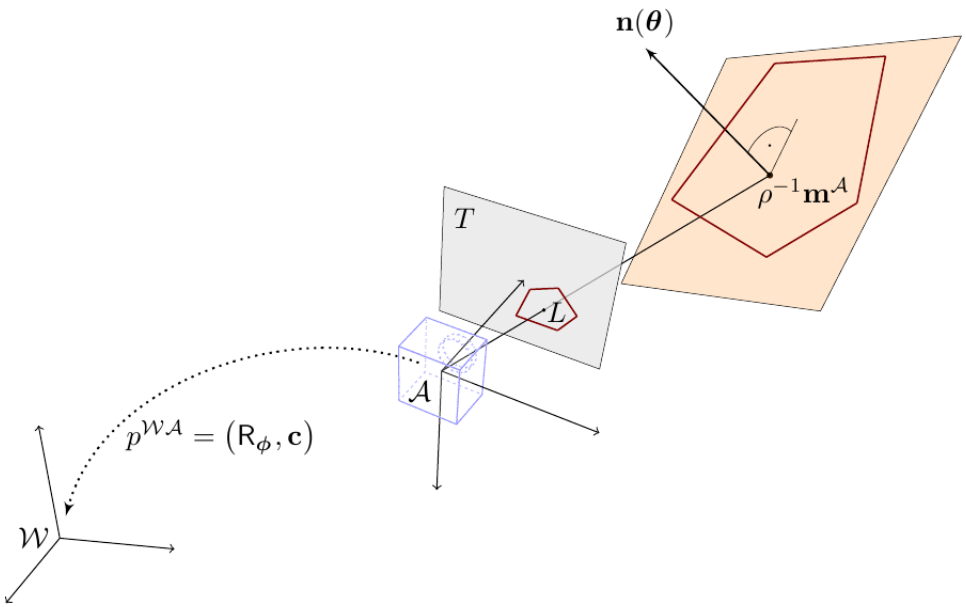


Figure 7.3.: Planar feature model. The pose of the anchor $\mathcal{A}$ with respect to the world reference frame $\mathcal{W}$ is given by translation $\mathbf{c}$ and rotation $\boldsymbol{\phi}$. The unit vector $\mathbf{m}^{\mathcal{A}}$ defines a ray in the anchor frame, and ρ is the inverse depth to the feature plane along this ray. The normal vector of the plane is given by $\mathbf{n}(\boldsymbol{\theta})$. Both, $\mathbf{n}(\boldsymbol{\theta})$ and $\mathbf{m}^{\mathcal{A}}$, are represented relative to the anchor frame. The template T is the camera image from the anchor pose. L is the outline of the feature in the template image. The back-projection of L onto the feature plane defines the shape of the planar segment that is represented by this feature.

image, i.e., the right camera image at the initial observation. The feature outline L describes a region of the reference image which corresponds to the projection of the feature. The shape of the planar feature in the scene is a back-projection of this outline onto the feature plane. The outline is fixed in the reference image which means that the estimated scene shape of the feature varies with the estimate of the feature plane position and normal.

The unit vector $\mathbf{m}^{\mathcal{A}}$ encodes a ray direction in the anchor frame $\mathcal{A}$. The ρ parameter measures the inverse depth to the intersection of this ray and the feature plane. That means the point $\rho^{-1}\mathbf{m}^{\mathcal{A}}$ lies on the plane. In principle, the choice of $\mathbf{m}^{\mathcal{A}}$ is arbitrary as long as it intersects the feature plane. In practice, we choose $\mathbf{m}^{\mathcal{A}}$ to go through the center of the feature outline. Just like with the viewpoint-based model, we model the ray $\mathbf{m}^{\mathcal{A}}$ as fixed with respect to the anchor frame. While the choice of $\mathbf{m}^{\mathcal{A}}$ is arbitrary, the important thing is that it is *chosen* instead of measured. There is no uncertainty involved and thus, it is a fixed parameter and not modeled as part of the probabilistic state vector.

The normal of the plane is also represented with respect to the anchor coordinate frame. The normal is represented in polar coordinates, i.e., by azimuth angle θ_x and elevation

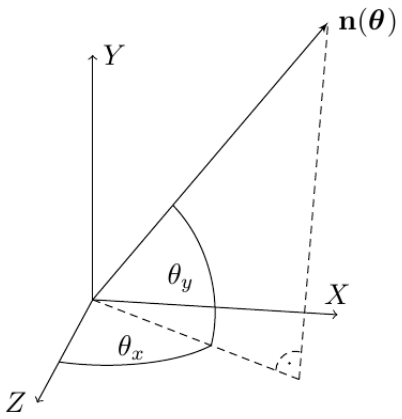


Figure 7.4.: Polar coordinates for the normal vector. The normal direction is encoded by azimuth angle θ_x and elevation angle θ_y.

angle θ_y, cf. Figure 7.4. The corresponding unit normal vector is

$$\mathbf{n}(\boldsymbol{\theta}) = \begin{pmatrix} \sin(\theta_x)\cos(\theta_y) \\ \sin(\theta_y) \\ \cos(\theta_x)\cos(\theta_y) \end{pmatrix}. \tag{7.2}$$

Following the same reasoning as with point feature bundles, the efficiency of the representation may be increased by sharing parameters among features. The representation of the anchor $\mathbf{c}, \boldsymbol{\phi}$ may be shared between planar features that are initialised in the same camera image. Following along the lines of Section 6.6, it is straightforward to make this extension and we will not discuss further details here.

7.4.2. Measurement Model

In this section, we introduce a generative measurement model for the planar features. For simplicity, we develop the measurement model for only a single feature $\mathbf{y}$. We omit the feature index to unclutter the notation. Extending the model to the measurement of several features is straightforward.

In the proposed model, the raw camera images are directly used as measurements. In the beginning we will keep the discussion strictly monocular, i.e., we restrict measurements to the right image of the stereo pair.[2] We denote the current image by $C : \mathbb{R}^2 \to \mathbb{R}$, i.e., as a function from pixel locations to intensities.

A measurement of a planar features consists of the observed intensities at a set of image pixels. Our generative model then must predict the intensity at a given pixel in the current image from given state parameters, i.e., from the current camera pose $\mathbf{x}_v$ and the feature state $\mathbf{y}$.

The basis for the measurement model is the homography transformation induced by the feature plane. We denote by $w^{\mathcal{CA}} : \mathbb{R}^2 \to \mathbb{R}^2$ the homography warp function which

[2] Pixels of the left image simply contribute additional measurements. Integrating these measurements is straightforward as we will see later.

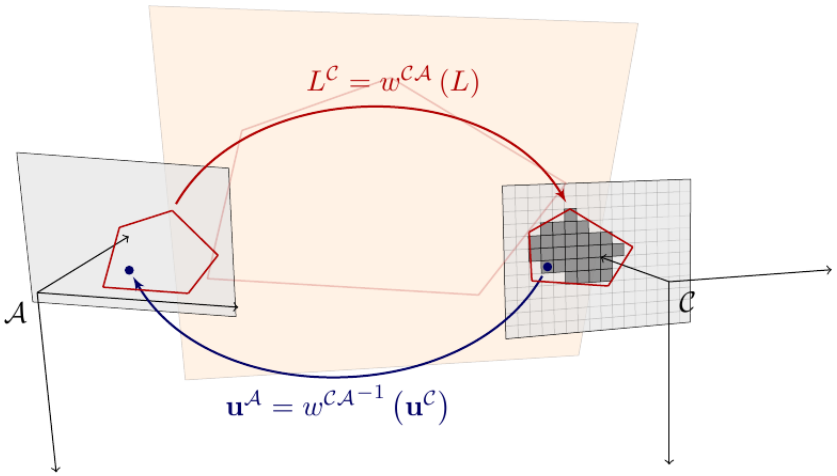


Figure 7.5.: The camera and feature state define a homography warp $w^{\mathcal{CA}}$ between the template and current image. The predicted warp is used to project the feature outline into the current image (red arrow). The inverse of $w^{\mathcal{CA}}$ can be used to predict pixel intensities in the current image by looking up the intensities at the corresponding template pixels (blue arrow). The measurement consists of the observed intensities of pixels inside the feature outline in the current image.

maps points in the template image to the corresponding points in the current image, cf. Figure 7.5. This warp function is of course the same as was introduced in Section 6.7 for the appearance prediction of bundle features. Now, for planar features, it will become an integral part of the measurement model.

The warp function depends on the poses of the anchor (template) and the current camera as well as the pose of the plane. It was derived in Section 6.7.1, Equations 6.53 and 6.54 where it was parameterised as

$$w^{\mathcal{CA}}\left(\,\cdot\,;\mathbf{r}^{\mathcal{WC}},\mathbf{q}^{\mathcal{WC}},\mathbf{c},\boldsymbol{\phi},\rho,\mathbf{m}^{\mathcal{A}},\mathbf{n}^{\mathcal{A}}\right) \tag{7.3}$$

The parameters comprise the pose of the current camera $(\mathbf{r}^{\mathcal{WC}},\mathbf{q}^{\mathcal{WC}})$, the reference anchor pose $(\mathbf{c},\boldsymbol{\phi})$, as well as the pose of the feature plane $(\rho,\mathbf{m}^{\mathcal{A}},\mathbf{n}^{\mathcal{A}})$. Here, $\mathbf{n}^{\mathcal{A}}$ denotes the normal vector of the plane. For planar features, the normal is now contained in the feature state as polar coordinates $\boldsymbol{\theta}$, i.e., $\mathbf{n}^{\mathcal{A}} = \mathbf{n}(\boldsymbol{\theta})$. In the following, instead of (7.3) we briefly write

$$w^{\mathcal{CA}}\left(\,\cdot\,;\mathbf{x}\right), \tag{7.4}$$

meaning that the appropriate camera and feature state components are selected from the full state vector $\mathbf{x}$.

A measurement now takes the form of a vector of pixel intensities. Using the warp function parameterised on the prior state estimate, $w^{\mathcal{CA}}\left(\,\cdot\,;\overline{\boldsymbol{\mu}}\right)$, we can project the feature outline into the current image to determine where we expect to observe the feature. Let $\mathbf{U}^{\mathcal{C}} = \{\mathbf{u}_1^{\mathcal{C}},\ldots,\mathbf{u}_m^{\mathcal{C}}\}$ be the set of integer pixel locations inside the expected projected

outline. The measurement vector $\mathbf{z} = (z_1, \ldots, z_m)^\top$ consists of the intensities observed at these locations in the current image, i.e.,

$$z_j = C(\mathbf{u}_j^{\mathcal{C}}) \quad \text{for } j \in \{1, \ldots, m\}. \tag{7.5}$$

To formulate a generative model, we have to define the function $\mathbf{h}(\mathbf{x}, \boldsymbol{\delta})$, i.e., we have to specify what we expect to measure given a certain camera and feature state. The intensity we expect to measure at a given pixel in the current image is the intensity at the corresponding location in the template image. Given the pixel coordinate $\mathbf{u}_j^{\mathcal{C}}$ in the current image, the corresponding template coordinate is computed using the inverse of the warp function,

$$\mathbf{u}_j^{\mathcal{A}} = w^{\mathcal{C}\mathcal{A}^{-1}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{x}\right). \tag{7.6}$$

The inverse of the warp function is obtained in a straightforward way, using the inverse of the homography matrix $\mathsf{H}^{\mathcal{C}\mathcal{A}}$ and wrapping it with the projection to and from homogeneous coordinates (similar to Equation 6.53, page 117)

$$w^{\mathcal{C}\mathcal{A}^{-1}}(\,\cdot\,;\mathbf{x}) \;:\; \mathbf{u}^{\mathcal{C}} \mapsto h\left(\mathsf{H}^{\mathcal{C}\mathcal{A}^{-1}} h^{-1}\left(\mathbf{u}^{\mathcal{C}}\right)\right). \tag{7.7}$$

Looking up the intensity of the template image at $\mathbf{u}_j^{\mathcal{A}}$ gives the predicted intensity in the current image:

$$h_j(\mathbf{x}) = T\left(\mathbf{u}_j^{\mathcal{A}}\right) = T\left(w^{\mathcal{C}\mathcal{A}^{-1}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{x}\right)\right). \tag{7.8}$$

In general, $\mathbf{u}_j^{\mathcal{A}}$ will not coincide with an integer pixel position. We use bilinear interpolation to compute the template intensities for non-integer coordinates.

We assume that in the measured intensity, there is additive camera noise δ_j. We model $\delta_j \sim \mathcal{N}(0, r_j)$ as zero-mean Gaussian with variance r_j, and assume it is uncorrelated between pixel locations. The generative model for a single pixel is then

$$z_j = h_j(\mathbf{x}) + \delta_j = T\left(w^{\mathcal{C}\mathcal{A}^{-1}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{x}\right)\right) + \delta_j. \tag{7.9}$$

By stacking pixel measurements (7.9) into a vector we obtain the full measurement model

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \begin{pmatrix} T\left(w^{\mathcal{C}\mathcal{A}^{-1}}\left(\mathbf{u}_1^{\mathcal{C}}\,;\mathbf{x}\right)\right) \\ \vdots \\ T\left(w^{\mathcal{C}\mathcal{A}^{-1}}\left(\mathbf{u}_m^{\mathcal{C}}\,;\mathbf{x}\right)\right) \end{pmatrix} + \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_m \end{pmatrix}. \tag{7.10}$$

Measurements in the left image Remember, that we employ a stereo camera and thus can make additional measurements in the left image of the stereo pair. Conceptually there is not a great difference between a monocular and a stereo setting. The additional intensity measurements obtained from the left image contribute additional pixel measurements with a different warp function which takes into account the baseline offset of the left eye from the camera center. The warp function $w^{\mathcal{C}'\mathcal{A}}$ maps coordinates from the template image to the left current camera image at pose $\mathcal{C}'$. It was derived in Section 6.7.1, Equations 6.57 and 6.58.

Using $w^{\mathcal{C}'\mathcal{A}}$ we can follow through the same procedure as explained for the right eye above. The feature outline is projected into the left image C' using the warp parameterised on the prior state estimate. Pixel locations inside this projected outline provide the additional measurement locations in the left image. These pixel measurements are modeled analogously to (7.9) as

$$z'_j = h'_j(\mathbf{x}) + \delta_j = T\left(w^{\mathcal{C}'\mathcal{A}^{-1}}\left(\mathbf{u}_j^{\mathcal{C}'}\ ;\mathbf{x}\right)\right) + \delta_j. \tag{7.11}$$

They appear as additional rows in the stacked measurement model (7.10).

Measurements in the left eye play an important role especially during the first few observations of freshly initialised planar features. After initialisation, uncertainty in the feature normal is large. Left eye measurements render the normal directly observable from a single stereo image. Indeed, in our experiments the estimated normal vector in most cases is estimated correctly from the first observation of a feature.

7.4.3. Warp Re-Parameterisation and Measurement Model Jacobians

A planar feature measurement typically consists of thousands of individual pixel measurements, which makes the computation of the Jacobian matrix expensive. Nevertheless, the cost of computing the Jacobian for all the individual measurements can be mitigated by careful implementation. We will present a few details here to illustrate this. The derivation of the full Jacobians is postponed Appendix A. The following discussion focuses on the right image of the stereo pair. Equivalent considerations can be made for the left image.

The computation of the warp functions (and Jacobians) for an individual pixel can be broken down into two parts. The first part varies and has to be computed for every pixel. The second part is constant across all pixels and thus needs to be computed only once. Computing the second part is usually expensive, while the first part is comparatively simple. Similar warp re-parameterisation methods were used by Hager & Belhumeur (1998) and Molton et al. (2004b).

In the case of our measurement model we split the measurement function h_j, Equation 7.8, into the computation of the homography $\mathsf{H}^{\mathcal{C}\mathcal{A}}$ from the state parameters, and a general homography warping and lookup function. The homography matrix $\mathsf{H}^{\mathcal{C}\mathcal{A}}$ is a parameter to the general homography warp. To make it easy to split the Jacobian, it is useful to pass the matrix as a parameter vector. To this end, we define the following functions, which stack the entries of a 3×3 matrix into a 9-vector, and unstack the vector into a matrix respectively:

$$S(\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}) = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{pmatrix} \qquad S^{-1}(\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{pmatrix}) = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}. \tag{7.12}$$

We define the general homography warp function

$$w^{\mathsf{H}}(\,\cdot\,;\mathbf{p}) \;:\; \mathbb{R}^2 \to \mathbb{R}^2 \;:\; \mathbf{u} \mapsto h\left(S^{-1}(\mathbf{p}) \cdot h^{-1}(\mathbf{u})\right), \tag{7.13}$$

parametrised with the 9-vector $\mathbf{p}$ which is a stacked homography matrix. The w^{H} function unstacks the homography and applies it to its argument (with the usual projection to and from homogeneous coordinates).

Using this general warp we can write Equation 7.8 as

$$h_j(\mathbf{x}) = T\left(\mathbf{u}_j^{\mathcal{A}}\right) = T\left(w^{\mathsf{H}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{p}(\mathbf{x})\right)\right) \tag{7.14}$$

with

$$\mathbf{p}(\mathbf{x}) = S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}^{-1}}(\mathbf{x})\right) \tag{7.15}$$

The warp parameter vector $\mathbf{p}(\mathbf{x})$ is independent of the pixel location and has to be computed only once for all pixels. It only depends on the state of the camera and the feature. We have added the state vector $\mathbf{x}$ as an argument to the homography matrix $\mathsf{H}^{\mathcal{C}\mathcal{A}^{-1}}(\mathbf{x})$ and the parameter vector $\mathbf{p}(\mathbf{x})$ to make this dependence explicit.

Using the chain rule, the Jacobian for the individual pixel measurement is split accordingly.

$$\frac{\partial h_j(\mathbf{x})}{\partial \mathbf{x}} = \frac{\partial T\left(\mathbf{u}_j^{\mathcal{A}}\right)}{\partial \mathbf{u}_j^{\mathcal{A}}} \cdot \frac{\partial w^{\mathsf{H}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{p}\right)}{\partial \mathbf{p}} \cdot \frac{\partial \mathbf{p}(\mathbf{x})}{\partial \mathbf{x}} \tag{7.16}$$

$$= \underbrace{\nabla T\left(\mathbf{u}_j^{\mathcal{A}}\right) \cdot \frac{\partial w^{\mathsf{H}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{p}\right)}{\partial \mathbf{p}}}_{\mathsf{H}_{T;j}} \cdot \underbrace{\frac{\partial \mathbf{p}(\mathbf{x})}{\partial \mathbf{x}}}_{\mathsf{H}_w} \tag{7.17}$$

We have now written the Jacobian of h_j as the product of two matrices $\mathsf{H}_{T;j}$ and H_w. The factor H_w is the Jacobian of the warp parameters by the state vector.[3] H_w is independent of the pixel location $\mathbf{u}_j^{\mathcal{C}}$. It is constant across all h_j for this feature and has to be computed only once.

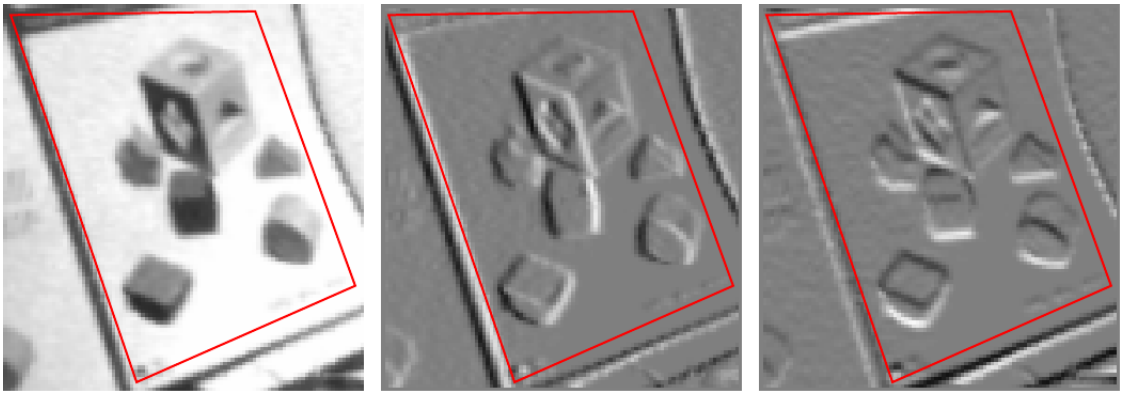
The first factor $\mathsf{H}_{T;j}$ is the Jacobian of the image intensity by the warp parameters.[4] It can be further split into the template gradient ∇T evaluated at the warped position $\mathbf{u}_j^{\mathcal{A}}$, and the derivative of the warped position by the homography parameters, evaluated at the j-th current image pixel $\mathbf{u}_j^{\mathcal{C}}$. The template gradient

$$\nabla T(x,y) = \left(\tfrac{\partial T(x,y)}{\partial x}, \quad \tfrac{\partial T(x,y)}{\partial y}\right) \tag{7.18}$$

is illustrated in Figure 7.6. $\mathsf{H}_{T;j}$ must be evaluated at every pixel. However, in comparison to H_w this computation is very simple.

[3] H_w is a 9×16 matrix. It contains the partial derivatives of the 9 entries of the homography matrix by the 16 relevant state entries (current camera pose and feature state).

[4] $\mathsf{H}_{T;j}$ is a 1×9 matrix, containing the partial derivatives of the image intensity at $\mathbf{u}_j^{\mathcal{A}}$ by the 9 entries of the homography matrix.



a) b) c)

Figure 7.6.: a) The template image T for one planar feature. b) the image derivative in X, and c) the image derivative in Y direction, i.e., the components of ∇T. The feature outline L is overlaid for reference.

Finally, note that the matrix product (7.17) need not be evaluated for every pixel measurement. This can be folded into the EKF update equations instead, by reordering matrix multiplications in the computations of the innovation covariance S and the gain matrix K. However, because we will not use the EKF update directly we will not go into detail here. We will revisit the issue in Section 7.8.1.

For the left-eye measurements, the warp and its Jacobian can be broken down analogously. The only difference lies in the warp parameter vector $\mathbf{p}(\mathbf{x})$, which in this case comprises the homography from the left image to the template

$$\mathbf{p}(\mathbf{x}) = S\big(\mathsf{H}^{\mathcal{C}'\mathcal{A}^{-1}}(\mathbf{x})\big). \tag{7.19}$$

The detailed Jacobians H_w and $\mathsf{H}_{T;j}$ are derived in Appendix A.8 for both, the right and the left image.

7.5. Fusing Measurement Information

For planar feature measurements, the dimension of the measurement vector $\mathbf{z}$ depends on the number of pixels in the area where the planar feature is visible. This will lead to serious performance problems if the model is applied straightforwardly in the update. In principle, we could simply collect those measurements into a large measurement vector and use it directly to update the state. However, in addition to the quadratic complexity of the update step in the size of the state vector, we have to deal with a cubic complexity in the size of the measurement vector. In the EKF update we need to invert the innovation covariance, which is a matrix of the same dimensions as the measurement vector. This is an $\mathcal{O}(m^3)$ operation in the dimension m of the measurement vector.[5] For point features,

[5]Complexity of computing the matrix inverse is $\mathcal{O}(m^3)$ using standard methods, such as Gaussian elimination. Algorithms with lower complexity exist. The asymptotically fastest currently known algorithm,

the measurement vector is short and the cubic complexity is not of concern. However, for more complex landmarks this becomes an issue.

Typically, a measurement of a planar feature comprises *thousands* of pixels. In a real-time system, inverting several 1000×1000 matrices many times per second is out of the question. The dimension of the measurement must be reduced. Preferably, this should be done without losing information.

It should be noted that a solution to this problem is of broader interest. In particular in SLAM, we can expect that a reliable estimation of the geometric parameters of complex landmarks such as line segments or planar surface patches requires high-dimensional measurements. By definition, such landmarks provide a high degree of abstraction, i.e., they compactly represent larger parts of the scene. Their parameters are usually not directly observed by sensors. More commonly, a single measurement of such a feature consists of a larger set of "raw" sensor measurements such as image intensities or points in a laser scan. Therefore, ideally, we want to avoid hand-crafted solutions and derive a principled, generally applicable way of handling high-dimensional measurements in the EKF framework. Thus, we will formalise the problem within a more general setting in the next section.

7.5.1. General Problem Statement

Assume we make a measurement $\mathbf{z}_\alpha$ of a state $\mathbf{x}$. Let the state space be partitioned into two parts

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_\alpha \\ \mathbf{x}_\beta \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_\alpha \\ \boldsymbol{\mu}_\beta \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\alpha\alpha} & \boldsymbol{\Sigma}_{\alpha\beta} \\ \boldsymbol{\Sigma}_{\beta\alpha} & \boldsymbol{\Sigma}_{\beta\beta} \end{bmatrix} \tag{7.20}$$

such that the measurement is a function of the sub-state $\mathbf{x}_\alpha$ only. We assume that $\dim(\mathbf{x}_\alpha)$ is bounded by a constant, where $\dim(\cdot)$ denotes the dimension of a vector. This assumption is equivalent to the assumption commonly made in SLAM, that the number of features measured at every time-step is limited.

In the case of planar features, the sub-state $\mathbf{x}_\alpha$ consists of the camera state $\mathbf{x}_v$ and the states $\mathbf{y}_i$ of the features being measured. Without loss of generality we can assume that state variables have been appropriately reordered.

The measurement can now be modelled as

$$\mathbf{z}_\alpha = \mathbf{h}\left(\mathbf{x}_\alpha\right) + \boldsymbol{\delta} \tag{7.21}$$

where $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \mathsf{R})$ is zero-mean Gaussian measurement noise with covariance R. Note that we assume additive noise instead of modelling $\boldsymbol{\delta}$ more generally as a parameter of $\mathbf{h}$ (as we did before, cf. Equation 3.56, for example). This is done purely for clarity of presentation. Nonlinear noise dependencies can be easily linearised to give a model of the form (7.21).

---

the Coppersmith–Winograd algorithm, has complexity $\mathcal{O}(m^{2.376})$. However, it is of purely theoretic interest because the asymptotic advantage only shows for matrices of astronomic dimensions (Bürgisser, 1996). Other algorithms, such as the famous Strassen algorithm (Strassen, 1969) with complexity $\mathcal{O}(m^{2.807})$ are also used in practice, although at the price of reduced numerical stability.

Consequently, it is difficult to give a complexity for "the" matrix inversion. For the purposes of this thesis we take the liberty to stick with $\mathcal{O}(m^3)$ because this is the complexity of many practical implementations. The cautious reader may substitute $\mathcal{O}(m^{2.3})$ for that.

Furthermore, we assume that the measurement consists of a large number of conditionally independent components, That is, the measurement function is of the form

$$\mathbf{h}\left(\mathbf{x}_\alpha\right) = \begin{pmatrix} h_1\left(\mathbf{x}_\alpha\right) \\ \vdots \\ h_m\left(\mathbf{x}_\alpha\right) \end{pmatrix} \tag{7.22}$$

where $m \gg \dim(\mathbf{x}_\alpha)$. Similarly, the components of the noise vector are assumed independent, i.e., the measurement noise covariance matrix R is a block-diagonal.

$$\mathsf{R} = \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_m \end{bmatrix} \tag{7.23}$$

The measurement and measurement model (7.21) are split accordingly.

$$\mathbf{z} = \begin{pmatrix} z_1 \\ \vdots \\ z_m \end{pmatrix} = \begin{pmatrix} h_1\left(\mathbf{x}_\alpha\right) + \delta_1 \\ \vdots \\ h_m\left(\mathbf{x}_\alpha\right) + \delta_m \end{pmatrix}. \tag{7.24}$$

Given the above structure, the components z_j are conditionally independent given $\mathbf{x}_\alpha$. To simplify the following discussion, we assume that z_j, $h_j(\mathbf{x}_\alpha)$, and δ_j are scalars, i.e., the number of independent components m equals the dimension of the measurement vector. For the planar measurements this assumption is true: every pixel contributes a scalar intensity measurement. In general, this is not necessary. The result presented in this section applies to any model with the above block structure.

The model can be used to update the state estimate with information from a measurement using the EKF update equations (cf. Algorithm 7). Given a prior estimate of the state $\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)$ and a measurement, the EKF computes the posterior estimate $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as

$$\mathsf{S} = \mathsf{H}\overline{\boldsymbol{\Sigma}}\mathsf{H}^\top + \mathsf{R} \tag{7.25}$$

$$\mathsf{K} = \overline{\boldsymbol{\Sigma}}\mathsf{H}^\top\mathsf{S}^{-1} \tag{7.26}$$

$$\boldsymbol{\mu} = \overline{\boldsymbol{\mu}} + \mathsf{K}\boldsymbol{\nu} \tag{7.27}$$

$$\boldsymbol{\Sigma} = \overline{\boldsymbol{\Sigma}} - \mathsf{K}\mathsf{H}\overline{\boldsymbol{\Sigma}} \tag{7.28}$$

where the innovation is the difference between the actual and predicted measurement

$$\boldsymbol{\nu} = \mathbf{z} - \mathbf{h}(\overline{\boldsymbol{\mu}}) \tag{7.29}$$

and we denote by H the Jacobian matrix of partial derivatives of the measurement function by the state

$$\mathsf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}. \tag{7.30}$$

The matrix H has $\dim(\mathbf{z}) = m$ rows and $\dim(\mathbf{x})$ columns. Because $\mathbf{h}$ only depends on the $\mathbf{x}_\alpha$ part of the state, the columns corresponding to $\mathbf{x}_\beta$ are zero.

$$\mathsf{H} = \frac{\partial \mathbf{h}\left(\mathbf{x}_\alpha\right)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}_\alpha)}{\partial \mathbf{x}_\alpha} & \mathbf{0} \\ \vdots & \vdots \\ \frac{\partial h_m(\mathbf{x}_\alpha)}{\partial \mathbf{x}_\alpha} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathsf{H}^\alpha & \mathbf{0}. \end{bmatrix} \tag{7.31}$$

If the dimension of the measurement vector is high the following becomes a concern: The innovation covariance S (Equation 7.25) is a dense $m \times m$ matrix, despite the sparse structure of H and R. Evaluating the gain matrix K requires the inversion of this matrix. This implies the cubic cost in m of the EKF update.

The sparse structure of H or R is of no immediate help here. However, having $m \gg \dim(\mathbf{x}_\alpha)$ independent measurements of $\mathbf{x}_\alpha$ suggests that there must be redundancy in those measurements. Consequently, the question should be asked: How can the information in the measurement be expressed more compactly? That is, how can the dimension of the measurement be reduced without losing information?

To express this question formally we need to define the notions of *measurement* and *equivalence of measurements.*

Definition 7.5.1 (Measurement)**.** *A measurement M is a triple $(\boldsymbol{\nu}, \mathsf{H}, \mathsf{R})$ of innovation, Jacobian, and noise covariance, i.e., the measurement-related quantities occurring in the EKF update equations.*

The dimension of the measurement is denoted as $\dim(M)$. The dimension of the measurement is reflected in the number of rows in $\boldsymbol{\nu}$, the number of rows in H, and the number of rows and columns in R.

An equivalence relation on measurements with respect to a prior state estimate $\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)$ can be defined as follows.

Definition 7.5.2 (Measurement Equivalence)**.** *Two measurements M and M' are called equivalent with respect to the prior state estimate $\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)$ if they yield the same posterior after an update step. We write this as $M \equiv M'\big|_{\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)}$.*

Note, that the equivalence of two measurements does not imply that they have the same dimension.

The problem we would like to solve can now be posed as follows: Given prior state estimate $\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)$ and measurement $M = (\boldsymbol{\nu}, \mathsf{H}, \mathsf{R})$ of the form discussed above,[6] find a *fused measurement* M_F such that

1.) $M_F \equiv M\big|_{\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)}$, and

2.) $\dim(M_F) \ll \dim(M)$.

In the following, we show that we can always find M_F such that $\dim(M_F) \leq \dim(\mathbf{x}_\alpha)$, i.e., the dimension of M_F is bounded by a constant. Furthermore, the cost of computing M_F depends linearly on $\dim(M)$.

This implies a reduction of the complexity of the EKF update step from $\mathcal{O}(m^3)$ in the size of the measurement vector to $\mathcal{O}(m)$.

[6] that is, depending only on sub-state $\mathbf{x}_\alpha$ and being composed of conditionally independent components.

7.5.2. Fusing High-Dimensional Measurements

Restriction to Sub-State

First, note that the task can be restricted to the directly observed sub-state. Because $\mathbf{h}$ only directly depends on $\mathbf{x}_\alpha$, estimates of quantities in $\mathbf{x}_\beta$ are updated only via existing correlations.

It is obvious that whenever two such measurements are equivalent with respect to the sub-state $\mathbf{x}_\alpha$ they are equivalent with respect to the full state as well (assuming the Jacobian columns corresponding to $\mathbf{x}_\beta$ are appropriately padded with zeros). Observing that the marginal distribution over $\mathbf{x}_\alpha$ is represented by blocks $\boldsymbol{\mu}_\alpha$, $\boldsymbol{\Sigma}_{\alpha\alpha}$ in Equation 7.20, we can write this formally as follows.

Proposition 7.5.3. *Let $M = (\boldsymbol{\nu}, \mathsf{H}^\alpha, \mathsf{R})$ and $M' = \left(\boldsymbol{\nu}', \mathsf{H}^{\alpha\prime}, \mathsf{R}'\right)$ be measurements that depend on sub-state $\mathbf{x}_\alpha$ only. That is, the measurement model is of the form of Equation 7.21. Then*

$$(\boldsymbol{\nu}, \mathsf{H}^\alpha, \mathsf{R}) \equiv \left(\boldsymbol{\nu}', \mathsf{H}^{\alpha\prime}, \mathsf{R}'\right)\big|_{(\overline{\boldsymbol{\mu}}_\alpha, \overline{\boldsymbol{\Sigma}}_{\alpha\alpha})} \implies \left(\boldsymbol{\nu}, \begin{bmatrix}\mathsf{H}^\alpha & \mathbf{0}\end{bmatrix}, \mathsf{R}\right) \equiv \left(\boldsymbol{\nu}', \begin{bmatrix}\mathsf{H}^{\alpha\prime} & \mathbf{0}\end{bmatrix}, \mathsf{R}'\right)\big|_{(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}})}.$$

This is easily verified by writing out the EKF update equations in detail.

Because of Proposition 7.5.3, it suffices to find a fused measurement M_F which is equivalent to $(\boldsymbol{\nu}, \mathsf{H}^\alpha, \mathsf{R})$ with respect to $\left(\overline{\boldsymbol{\mu}}_\alpha, \overline{\boldsymbol{\Sigma}}_{\alpha\alpha}\right)$. Consequently, we focus the discussion on the sub-state $\mathbf{x}_\alpha$. For convenience we omit the α subscripts. In the following, $\boldsymbol{\mu}$ should be understood as referring to $\boldsymbol{\mu}_\alpha$ and so on, unless otherwise noted.

Canonical Representation and Information Filter Update

To approach the problem, a change of perspective is useful. Instead of representing a Gaussian probability density function by first- and second-order moments, it can equivalently be characterised in the canonical representation, which is also referred to as the information form. Interestingly, in the canonical representation the update becomes linear in the measurement dimension m. This occurs because the total information provided by the measurement $\mathbf{z}$ can be naturally expressed as the sum of the information contributed by the m independent components z_j.

The canonical representation of a multivariate Gaussian is given by an information matrix $\boldsymbol{\Omega}$ and information vector $\boldsymbol{\xi}$, which are related to mean and covariance as

$$\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1} \tag{7.32}$$

$$\boldsymbol{\xi} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \tag{7.33}$$

The Extended Information Filter (EIF) is the canonical equivalent of the EKF. A detailed presentation of the EIF can be found in (Thrun et al., 2005, section 3.5.4). The update step, which performs exactly the same operation as the EKF update, is given by

$$\Delta\boldsymbol{\xi} = \mathsf{H}^\top \mathsf{R}^{-1} \left(\boldsymbol{\nu} + \mathsf{H}\overline{\boldsymbol{\mu}}\right) \tag{7.34}$$

$$\Delta\boldsymbol{\Omega} = \mathsf{H}^\top \mathsf{R}^{-1} \mathsf{H} \tag{7.35}$$

$$\boldsymbol{\xi} = \overline{\boldsymbol{\xi}} + \Delta\boldsymbol{\xi} \tag{7.36}$$

$$\boldsymbol{\Omega} = \overline{\boldsymbol{\Omega}} + \Delta\boldsymbol{\Omega} \tag{7.37}$$

Because of the diagonal structure of the measurement noise covariance (7.23) the information update can be written as a sum:

$$\Delta\boldsymbol{\xi} = \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} \left(\nu_j + \mathsf{H}_j \overline{\boldsymbol{\mu}}\right) \tag{7.38}$$

$$\Delta\boldsymbol{\Omega} = \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} \mathsf{H}_j. \tag{7.39}$$

In the above equations, H_j denotes the j-th row of the measurement Jacobian, and ν_j denotes the j-th element of the innovation vector

$$\nu_j = z_j - h_j(\overline{\boldsymbol{\mu}}), \tag{7.40}$$

$$\mathsf{H}_j = \frac{\partial h_j(\mathbf{x})}{\partial \mathbf{x}}. \tag{7.41}$$

The summands in (7.39) are square matrices of bounded size[7] that can each be computed in constant time. The cost of evaluating the sum is linear in m. An analogous observation can be made for the sum in (7.38). Thus, the complexity of the EIF update step is $\mathcal{O}(m)$ in the dimension of the measurement.

A reader familiar with image alignment will recognise the similarity of these sums (7.38) and (7.39) to terms from the least squares solution in the Lucas-Kanade method (Baker & Matthews, 2004). This is interesting insofar as the application we have in mind, i.e., intensity measurements of planar features, essentially is solving an image alignment problem. Indeed, the Lucas-Kanade equations can be viewed as implementing an EIF update with a uniform prior. The probabilistic image alignment algorithm proposed by Molton et al. (2004b) can be viewed as approximating an Iterated Extended Information Filter.

Decomposing the Information Update

We return to the idea of reducing the dimension of the original measurement to transfer this efficiency to the EKF framework. An equivalence relation $\equiv_c$ of measurements with respect to a prior in information form can be defined analogously to Definition 7.5.2.

Definition 7.5.4 (Measurement Equivalence (Information Form))**.** *Two measurements M and M' are called equivalent with respect to the prior state estimate $\left(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\Omega}}\right)$ if they yield the same posterior $(\boldsymbol{\xi}, \boldsymbol{\Omega})$ after an update step. We write this as $M \equiv_c M'\big|_{\left(\overline{\boldsymbol{\xi}}, \overline{\boldsymbol{\Omega}}\right)}$.*

Because EKF and EIF update implement the same operation, both notions are obviously equivalent, i.e.,

$$M \equiv M'\big|_{\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}\right)} \iff M \equiv_c M'\big|_{\left(\overline{\boldsymbol{\Sigma}}^{-1}\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}^{-1}\right)}. \tag{7.42}$$

Equations 7.36 and 7.37 for computing the posterior imply that two measurements are equivalent if and only if they result in the same information update $\Delta\boldsymbol{\xi}$, $\Delta\boldsymbol{\Omega}$. Applied

[7] Remember, that we discuss only the sub-state $\mathbf{x}_\alpha$ here, with dimension bounded by a constant.

to the problem at hand, this means that in order to find the compact fused measurement $M_F = (\boldsymbol{\nu}_F, \mathsf{H}_F, \mathsf{R}_F)$ we have to solve the following system of equations

$$\mathsf{H}_F^\top \mathsf{R}_F^{-1} (\boldsymbol{\nu}_F + \mathsf{H}_F \overline{\boldsymbol{\mu}}) = \Delta\boldsymbol{\xi} = \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} (\nu_j + \mathsf{H}_j \overline{\boldsymbol{\mu}}) \tag{7.43}$$

$$\mathsf{H}_F^\top \mathsf{R}_F^{-1} \mathsf{H}_F = \Delta\boldsymbol{\Omega} = \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} \mathsf{H}_j. \tag{7.44}$$

A solution can be found from the eigen-decomposition of $\Delta\boldsymbol{\Omega}$.[8] The matrix $\Delta\boldsymbol{\Omega}$ is symmetric and positive semi-definite by construction. Thus, all its eigenvalues are real and either zero or positive. Its eigen-decomposition always exists and is of the form (Hartley & Zisserman, 2000)

$$\mathsf{U}\Lambda\mathsf{U}^\top = \Delta\boldsymbol{\Omega}. \tag{7.45}$$

The diagonal matrix Λ comprises the eigenvalues of $\Delta\boldsymbol{\Omega}$. The orthogonal columns of U are the corresponding eigenvectors. Both, U and Λ, are square matrices of the same dimension as $\Delta\boldsymbol{\Omega}$, i.e., $\dim(\mathbf{x}_\alpha) \times \dim(\mathbf{x}_\alpha)$.

Eigenvectors corresponding to zero eigenvalues can be understood as representing directions in the state space about which the measurement provides no information. They can be removed without changing $\Delta\boldsymbol{\Omega}$. Let U' be the matrix obtained by removing from U every column corresponding to a zero eigenvalue. Let Λ' be the matrix obtained by removing from Λ every row and column corresponding to a zero eigenvalue.[9] It is easily verified that

$$\mathsf{U}'\Lambda'\mathsf{U}'^\top = \mathsf{U}\Lambda\mathsf{U}^\top = \Delta\boldsymbol{\Omega}. \tag{7.46}$$

Furthermore, Λ' is a diagonal matrix with $\lambda_i > 0$ for the diagonal entries. Thus it is invertible and the inverse is positive semi-definite, i.e., a valid covariance matrix. Hence, we can choose

$$\mathsf{H}_F = \mathsf{U}'^\top \tag{7.47}$$

$$\mathsf{R}_F = \Lambda'^{-1}. \tag{7.48}$$

This satisfies Equation 7.44. It remains to find $\boldsymbol{\nu}_F$. From (7.43) we derive

$$\mathsf{H}_F^\top \mathsf{R}_F^{-1} (\boldsymbol{\nu}_F + \mathsf{H}_F \overline{\boldsymbol{\mu}}) = \Delta\boldsymbol{\xi} \tag{7.49}$$

$$\mathsf{H}_F \mathsf{H}_F^\top \mathsf{R}_F^{-1} (\boldsymbol{\nu}_F + \mathsf{H}_F \overline{\boldsymbol{\mu}}) = \mathsf{H}_F \Delta\boldsymbol{\xi} \tag{7.50}$$

$$\mathsf{R}_F^{-1} (\boldsymbol{\nu}_F + \mathsf{H}_F \overline{\boldsymbol{\mu}}) = \mathsf{H}_F \Delta\boldsymbol{\xi} \tag{7.51}$$

$$\boldsymbol{\nu}_F = \mathsf{R}_F \mathsf{H}_F \Delta\boldsymbol{\xi} - \mathsf{H}_F \overline{\boldsymbol{\mu}}. \tag{7.52}$$

[8] At first glance, an obvious idea seems to be the choice

$$\mathsf{H}_F = \mathbf{I}, \qquad \mathsf{R}_F = \Delta\boldsymbol{\Omega}^{-1}.$$

However, this fails in the general case because $\Delta\Omega$ may be singular. This will happen exactly if the measurement carries no information about some directions in the state space, which points the way to the correct solution.

[9] In practice, rows and columns can be removed if the corresponding eigenvalue is below some small threshold ϵ.

The step from (7.50) to (7.51) can be made because the columns of U' are formed by the pairwise orthogonal eigenvectors. Thus $\mathsf{H}_F\mathsf{H}_F^\top = \mathsf{U}'^\top\mathsf{U}' = \mathbf{I}$.

Note, that the size of $\Delta\boldsymbol{\xi}$ and $\Delta\boldsymbol{\Omega}$ is $\dim(\mathbf{x}_\alpha)$, *independent* of the dimension m of the measurement. This means, the cost of computing the fused measurement M_F from $\Delta\boldsymbol{\xi}$, $\Delta\boldsymbol{\Omega}$ is constant.

Furthermore, the dimension of the fused measurement is $\dim(M_F) \leq \dim(\mathbf{x}_\alpha)$. This is because H_F and R_F are obtained from the matrices U' and Λ' which in turn have at most $\dim(\mathbf{x}_\alpha)$. The exact dimension depends on the number of rows and columns that were removed from U, Λ corresponding to zero eigenvalues. The maximum size $\dim(\mathbf{x}_\alpha)$ is reached precisely if the decomposition has no zero eigenvalue.

Simplification of $\boldsymbol{\nu}_F$

We can substitute the sum (7.38) for $\Delta\boldsymbol{\xi}$ in Equation 7.52. It turns out that it is not necessary to evaluate the information vector update $\Delta\boldsymbol{\xi}$ fully. We can avoid the computation of the terms $\mathsf{H}_j\overline{\boldsymbol{\mu}}$ in the sum. Substituting (7.38) into (7.52), we obtain the following simplified expression for $\boldsymbol{\nu}_F$.

$$
\begin{aligned}
\boldsymbol{\nu}_F &= \mathsf{R}_F\mathsf{H}_F\Delta\boldsymbol{\xi} - \mathsf{H}_F\overline{\boldsymbol{\mu}} \\
&= \mathsf{R}_F\mathsf{H}_F\Big(\sum_{j=1}^{m}\mathsf{H}_j^\top r_j^{-1}\left(\nu_j + \mathsf{H}_j\overline{\boldsymbol{\mu}}\right)\Big) - \mathsf{H}_F\overline{\boldsymbol{\mu}} && (7.53)\\
&= \Big(\mathsf{R}_F\mathsf{H}_F\sum_{j=1}^{m}\mathsf{H}_j^\top r_j^{-1}\nu_j\Big) + \Big(\mathsf{R}_F\mathsf{H}_F\underbrace{\sum_{j=1}^{m}\mathsf{H}_j^\top r_j^{-1}\mathsf{H}_j}_{\Delta\boldsymbol{\Omega}=\mathsf{H}_F^\top\mathsf{R}_F^{-1}\mathsf{H}_F}\overline{\boldsymbol{\mu}}\Big) - \mathsf{H}_F\overline{\boldsymbol{\mu}} && (7.54)\\
&= \Big(\mathsf{R}_F\mathsf{H}_F\sum_{j=1}^{m}\mathsf{H}_j^\top r_j^{-1}\nu_j\Big) + \Big(\underbrace{\mathsf{R}_F\mathsf{H}_F\mathsf{H}_F^\top\mathsf{R}_F^{-1}}_{\mathbf{I}}\mathsf{H}_F\overline{\boldsymbol{\mu}}\Big) - \mathsf{H}_F\overline{\boldsymbol{\mu}} && (7.55)\\
&= \mathsf{R}_F\mathsf{H}_F\sum_{j=1}^{m}\mathsf{H}_j^\top r_j^{-1}\nu_j && (7.56)
\end{aligned}
$$

Summary and Complete Algorithm

Let us summarise the results we have obtained so far in this section. We have shown that given a measurement M with $\dim(M) = m$, we can compute an equivalent fused measurement M_F. The fused measurement has dimension $\dim(M_F) \leq \dim(\mathbf{x}_\alpha)$, i.e., bounded by a constant. Remember that the construction above has been with respect to the sub-state $\mathbf{x}_\alpha$. Hence, equivalence means that given the prior estimate $\left(\overline{\boldsymbol{\mu}}_\alpha, \overline{\boldsymbol{\Sigma}}_{\alpha\alpha}\right)$ we have

$$M_F \equiv_c M\big|_{\left(\overline{\boldsymbol{\Sigma}}_{\alpha\alpha}^{-1}\overline{\boldsymbol{\mu}}_\alpha,\, \overline{\boldsymbol{\Sigma}}_{\alpha\alpha}^{-1}\right)}. \tag{7.57}$$

by construction, which is equivalent to

$$M_F \equiv M\big|_{\left(\overline{\boldsymbol{\mu}}_\alpha,\, \overline{\boldsymbol{\Sigma}}_{\alpha\alpha}\right)}. \tag{7.58}$$

Algorithm 15: FuseMeasurements

Input: Set of measurements $\{(\nu_1, \mathsf{H}_1, r_1), \ldots, (\nu_m, \mathsf{H}_m, r_m)\}$
Output: Fused measurement $(\boldsymbol{\nu}_F, \mathsf{H}_F, \mathsf{R}_F)$

1 $\Delta\boldsymbol{\Omega} := \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} \mathsf{H}_j$

2 $\Delta\boldsymbol{\xi}^+ := \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} \nu_j$

3 $\mathsf{U}\Lambda\mathsf{U}^\top :=$ EigenDecomposition $(\Delta\boldsymbol{\Omega})$

4 remove columns i from U that correspond to eigenvalues $\lambda_i = 0$.

5 remove rows and columns i from Λ that correspond to eigenvalues $\lambda_i = 0$.

6 $\mathsf{H}_F := \mathsf{U}^\top$

7 $\mathsf{R}_F := \Lambda^{-1}$

8 $\boldsymbol{\nu}_F := \mathsf{R}_F \mathsf{H}_F \Delta\boldsymbol{\xi}^+$

9 **return** $(\boldsymbol{\nu}_F, \mathsf{H}_F, \mathsf{R}_F)$

Using Proposition 7.5.3, we conclude

$$M_F \equiv M\big|_{(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}})}, \tag{7.59}$$

i.e., the equivalence also holds for the full state.[10]

Thus, the fused measurement can be used instead of the original one in the full EKF update step, yielding an identical posterior estimate. Because of the bounded dimension of M_F, the cost of this EKF update is constant with respect to m. There is the additional cost of computing M_F, which is linear in m.

The computation of a fused measurement is summarised in Algorithm 15. The algorithm takes a set of m independent measurement components as input and returns a single fused measurement.

A modified EKF update procedure, including the computation of the fused measurement is given in Algorithm 16. Note, that the fused measurement computation in lines 1–5 of the algorithm is restricted to sub-state $\mathbf{x}_\alpha$. Thus, for the full state update the fused measurement Jacobian must be padded with zero columns (line 6).

7.5.3. Alternative Method Based on QR Factorisation

In this section, we explore connections to the method developed by Bayard & Brugarolas (2004, 2005). In their independently developed approach measurement compression is performed based on the QR factorisation of the Jacobian matrix. To illustrate the

[10] For the full state update, H_F must be padded with zero columns to account for $\mathbf{x}_\beta$, as discussed at the beginning of this section. Note, that adding columns to H_F does not alter the dimension of the measurement M_F.

Algorithm 16: FusedEkfUpdate

Input: Prior belief $\overline{\boldsymbol{\mu}} = \begin{pmatrix} \overline{\boldsymbol{\mu}}_\alpha \\ \overline{\boldsymbol{\mu}}_\beta \end{pmatrix}$, $\overline{\boldsymbol{\Sigma}} = \begin{bmatrix} \overline{\boldsymbol{\Sigma}}_{\alpha\alpha} & \overline{\boldsymbol{\Sigma}}_{\alpha\beta} \\ \overline{\boldsymbol{\Sigma}}_{\beta\alpha} & \overline{\boldsymbol{\Sigma}}_{\beta\beta} \end{bmatrix}$,
Measurements $\mathbf{z} = (z_1, \ldots, z_m)^\top$ only depending on $\mathbf{x}_\alpha$,
Noise covariance $\mathsf{R} = \text{block-diag}\,(r_1, \ldots, r_m)$

Output: Posterior belief $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

forall $1 \leq j \leq m$ **do**
  $\mathsf{H}_j := \left.\dfrac{\partial h_j(\mathbf{x}_\alpha)}{\partial \mathbf{x}_\alpha}\right|_{\mathbf{x}_\alpha = \overline{\boldsymbol{\mu}}_\alpha}$
  $\nu_j := z_j - h_j(\overline{\boldsymbol{\mu}}_\alpha)$
end
$(\boldsymbol{\nu}_F, \mathsf{H}_F, \mathsf{R}_F) := \text{FuseMeasurements}\left(\{(\nu_1, \mathsf{H}_1, r_1), \ldots, (\nu_m, \mathsf{H}_m, r_m)\}\right)$
$\mathsf{H} := \begin{bmatrix} \mathsf{H}_F & \mathbf{0} \end{bmatrix}$ `// pad with 0 cols for` $\mathbf{x}_\beta$
$\mathsf{S} := \mathsf{H}\overline{\boldsymbol{\Sigma}}\mathsf{H}^\top + \mathsf{R}_F$
$\mathsf{K} := \overline{\boldsymbol{\Sigma}}\mathsf{H}^\top \mathsf{S}^{-1}$
$\boldsymbol{\mu} := \overline{\boldsymbol{\mu}} + \mathsf{K}\boldsymbol{\nu}_F$
$\boldsymbol{\Sigma} := \overline{\boldsymbol{\Sigma}} - \mathsf{K}\mathsf{H}\overline{\boldsymbol{\Sigma}}$
return $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

relationship between the approaches, we briefly derive an alternative fused measurement formulation using their proposed QR technique. The derivation is done in the information form and in the context of the sub-state $\mathbf{x}_\alpha$ introduced before.

The method requires a preprocessing of the measurement equations to obtain unit variance for the measurement noise in every component and thus the identity matrix as the measurement noise covariance R. In the present scenario, this is done by pre-multiplying every pixel measurement with $r_j^{-\frac{1}{2}}$ to obtain

$$\mathsf{H}_j = r_j^{-\frac{1}{2}} \left.\frac{\partial h_j(\mathbf{x}_\alpha)}{\partial \mathbf{x}_\alpha}\right|_{\mathbf{x}_\alpha = \overline{\boldsymbol{\mu}}_\alpha} \tag{7.60}$$

$$\nu_j = r_j^{-\frac{1}{2}} \left(z_j - h_j(\overline{\boldsymbol{\mu}}_\alpha)\right) \tag{7.61}$$

after which ν_j are distributed with $\mathcal{N}(0, 1)$. With identity measurement noise covariance we obtain the information update

$$\Delta\boldsymbol{\Omega} = \mathsf{H}^\top \mathsf{H} \tag{7.62}$$

where H is the stacked full Jacobian with rows H_j. Now consider the QR factorisation of $\mathsf{H} = \mathsf{A}\mathsf{B}$ where A is orthogonal and B is upper triangular.[11]

[11] We use A and B to denote the decomposition because Q and R already have been used with a different meaning in this thesis.

This factorisation is of the form

$$\mathsf{H} = \begin{bmatrix}\mathsf{A}_1 & \mathsf{A}_2\end{bmatrix} \begin{bmatrix}\mathsf{B}_1 \\ \mathbf{0}\end{bmatrix} = \mathsf{A}_1\mathsf{B}_1. \tag{7.63}$$

Here, B_1 is a $\dim(\mathbf{x}_\alpha) \times \dim(\mathbf{x}_\alpha)$ upper triangular matrix and A_1 is a $\dim(\mathbf{x}_\alpha) \times m$ orthogonal matrix. Using the factorisation, a decomposition of (7.62) for the fused measurement can be written

$$\mathsf{H}'_F{}^\top \mathsf{R}'^{-1}_F \mathsf{H}'_F = \Delta\boldsymbol{\Omega} = \mathsf{H}^\top\mathsf{H} = \mathsf{B}_1^\top\mathsf{A}_1^\top\mathsf{A}_1\mathsf{B}_1 = \mathsf{B}_1^\top\mathsf{B}_1 \tag{7.64}$$

where we choose

$$\mathsf{H}'_F = \mathbf{I} \tag{7.65}$$

$$\mathsf{R}'_F = \left(\mathsf{B}_1^\top\mathsf{B}_1\right)^{-1} = \mathsf{B}_1^{-1}\mathsf{B}_1^{-\top} \tag{7.66}$$

By plugging H'_F, R'_F into Equation 7.52 and simplifying, we obtain

$$\boldsymbol{\nu}'_F = \mathsf{B}_1^{-1}\mathsf{A}_1^\top\boldsymbol{\nu}. \tag{7.67}$$

Now, the problem remains that B_1 might be singular. This will happen if the measurement carries no information about some directions in the state space $\mathbf{x}_\alpha$. We can solve this by pre-multiplying the measurement $(\boldsymbol{\nu}'_F, \mathsf{H}'_F, \mathsf{R}'_F)$ with B_1 which yields

$$\boldsymbol{\nu}_F = \mathsf{B}_1\boldsymbol{\nu}'_F = \mathsf{A}_1^\top\boldsymbol{\nu} \tag{7.68}$$

$$\mathsf{H}_F = \mathsf{B}_1\mathsf{H}'_F = \mathsf{B}_1 \tag{7.69}$$

$$\mathsf{R}_F = \mathsf{B}_1\mathsf{R}'_F\mathsf{B}_1^\top = \mathbf{I} \tag{7.70}$$

By plugging $(\boldsymbol{\nu}_F, \mathsf{H}_F, \mathsf{R}_F)$ and $(\boldsymbol{\nu}'_F, \mathsf{H}'_F, \mathsf{R}'_F)$ into the EKF update equations it is easily verified that they are equivalent.

Thus we have obtained an alternative formulation of computing a fused measurement, which is based on the technique proposed by Bayard & Brugarolas (2004). An advantage of this method is its higher numerical accuracy. Decomposing H directly avoids squaring its condition number by forming $\mathsf{H}^\top\mathsf{H}$. Furthermore, in our approach near-zero eigenvalues below a threshold are discarded which may results in loss of accuracy. No such step is required in the QR method.

However, our method is approximately twice as fast: For $m \gg \dim(\mathbf{x}_\alpha) = a$, the most expensive operation in our method is forming the sum (7.39). This requires $\approx a^2 m$ floating point operations. In (Bayard & Brugarolas, 2004) the QR factorisation is the most expensive step. It requires $\approx 2a^2 m$ floating point operations.[12] Moreover, computing the sum (7.39) is absolutely straightforward to parallelize, which is another point in favour of our method.

[12] For instance using the Fast Givens QR, algorithm 5.2.4, in (Golub & Van Loan, 1996) which requires $2a^2(m - \frac{a}{3})$ floating point operations.

7.5.4. Discussion

We have shown a method to reduce high-dimensional measurements to fused measurements carrying equivalent information but with dimension bounded by a constant a. Here, $a = \dim(\mathbf{x}_\alpha)$ is the dimension of the state sub-vector on which the measurement directly depends. In our application scenario, $\mathbf{x}_\alpha$ comprises the camera pose and the feature state, so $a = 16$ parameters in total. In this case and in similar scenarios we have both $a \ll m$ and $a \ll n$.

We have observed that, in the Information Filter framework, the linear formulation of the update step is natural. Thus, applying the Extended Information Filter (EIF) instead of the EKF should be considered as an option. However, measurement fusion in combination with the EKF update provides advantages over both the EKF with the original measurement and the EIF. Denoting by n the size of the state vector and by m the size of the measurement, we can compare the overall complexity of the filters as follows.

The drawback of the EIF is the prediction step which is cubic in the dimension of the state. The overall complexity of SLAM using the EIF is $\mathcal{O}(n^3 + m)$ (Thrun et al., 2004). For EKF SLAM, the overall complexity is $\mathcal{O}(n^2 m + m^3)$.

Using the fused measurement in the EKF reduces the measurement dimension m to the constant a. The additional cost of computing the fused measurement is linear in m. The overall complexity of EKF SLAM with fused measurements then is $\mathcal{O}(n^2 + m)$. Thus we achieve a reduction of complexity with respect to both the raw EKF and the EIF.

Although the present work has been motivated by a specific application, let us emphasize once more that the result is of wider importance. In previous work, the problem of high measurement dimensionality has often not become significant or has been dealt with in application-specific ways. Examples of the latter case are (Smith et al., 2006; Weingarten & Siegwart, 2006). The approach there is to define a suitable (and low-dimensional) measurement space. In the Visual-SLAM-with-Lines setting of Smith et al. (2006), a measurement consists of the image projections of the endpoints of a line. The parameters of a plane are used in (Weingarten & Siegwart, 2006). A measurement and associated uncertainty in this space is then obtained, by fitting a model to the "raw" measurements. Care has to be taken in both, the choice of the measurement space and the implementation of the fitting operation, to avoid information loss.

Our approach of fusing measurements is different in that the appropriate low-dimensional space is selected automatically and optimally based on the information contained in the raw measurements. Furthermore, this is done on a measurement-to-measurement basis which makes the method applicable to problems where it is hard to find a good general representation by hand.[13]

[13] For example, in the planar SLAM scenario we might choose as a hand-crafted measurement space the 8 independent ratios of the homography transformations parameters. However, situations may arise, where the observed pixel pattern is degenerate in a way that introduces additional degrees of freedom. In this case the homography can not be fully determined and the measurement fails. Our approach naturally adapts to such situations by reducing the dimension of the fused measurement.

7.6. Iterative Measurement Process

In the previous section we have seen that it is possible to efficiently handle the large measurement vectors of direct intensity measurements. In this section we discuss another problem that we face with direct measurements, namely the high nonlinearity of the measurement model. Our measurement model (cf. Equation 7.10) relates the current image to the system state, which involves expressing image intensity as a function of image position. In general, the relationship between pixel position in an image and intensity is highly nonlinear.

The EKF update step requires the linearisation of the measurement model. That is, Equation 7.10 is approximated by a first-order Taylor expansion around the prior state estimate. For every measured pixel position, this involves linearising the template image function T around the predicted corresponding template pixel position. The intensity in the neighbourhood of a template pixel is approximated using the template gradient at that pixel, as discussed in Section 7.4.3. In general, this approximation is valid only in the immediate neighbourhood of the template pixel. Consequently, the linearised model used in the EKF is valid only in a small region of state-space around the true state. Straightforward application of the EKF will therefore result in a severe limitation of the camera acceleration that can be tolerated, i.e., the system will only work when the predicted and observed image motion are similar. The problem of relying on the linearity of the image gradient in this way has been identified and discussed for instance about the direct approach of Jin et al. (2003).

We tackle the problem by iterating the EKF update step. We apply a variation of the UpdateIekf procedure presented in Section 3.11. The iterative re-linearisation and re-computation of the mean vector is interleaved with the computation of fused measurements. In the $(k+1)$th iteration, first we create fused feature measurements using a linearisation of the intensity measurement model around the posterior mean estimate $\boldsymbol{\mu}^k$ from the previous iteration. Then we apply the IEKF equations to compute the new posterior estimate $\boldsymbol{\mu}^{k+1}$.

More specifically, in every iteration the following steps are performed for all features that are predicted to be currently visible. First, for every feature the set of pixel measurement locations is obtained. This is achieved by projecting the feature outline into the current image using the warp parameterised on the posterior estimate from the previous iteration, $w^{\mathcal{CA}}\left(\,\cdot\,;\boldsymbol{\mu}^k\right)$. The projected outline is then rasterised to obtain the set of pixel locations $\mathbf{U}^{\mathcal{C}}$. Next, the measurement vector is obtained from the current image intensities at the measurement locations. The predicted measurement is obtained by back-projecting measurement locations into the template image and looking up intensities using bilinear interpolation. The measurement model (7.10) is linearised at $\boldsymbol{\mu}^k$ to compute the Jacobian. This process is performed for both the left and right current image to obtain the full measurement vector comprising pixel intensities in both images. Using the full measurement obtained in this way, a fused measurement is computed for the feature.

The fused measurements for all features are stacked into a single measurement vector. This is then used in the EKF update equations to evaluate the posterior estimate $\boldsymbol{\mu}^{k+1}$ for the current iteration. The whole process is repeated until the posterior estimate converges.

During rapid camera accelerations the initial *a priori* estimate can be quite far from the

true state, such that the predicted positions in the template image are so far from their true positions that the initial linearisation of the measurement model is useless. To guide the update process to the correct region of convergence the first iteration is performed using a restricted, point-feature-like measurement model. For the restricted model, the measurement consists only of the 2D image coordinates of the projection of the feature center. As the restricted model we use the view-point based feature model described in Section 6.5. This can be applied in a straightforward way, because the parameters for the view-point based model are a sub-set of those of the planar feature model. This 2D measurement is obtained by active search using template matching as described in Section 4.7.

The iterative measurement and update step outlined above is detailed in Algorithm 17. The input to the algorithm is the prior state estimate $\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}$, the current image C, and a set F of feature indices. F contains the features that are predicted to be currently visible. The algorithm calculates and returns the posterior belief $\boldsymbol{\mu}, \boldsymbol{\Sigma}$.

The first iteration is summarised by the function InitialUpdate in line 2. As described above it uses a reduced measurement model and point-feature-like active search. $\boldsymbol{\mu}^0$ is the resulting posterior mean estimate.

The iterations of the intensity measurement process occur in the main loop in lines 3–22. Every loop iteration consists of constructing fused measurements of all features in F and calculating the new posterior mean $\boldsymbol{\mu}^{k+1}$.

The inner loop in lines 4–16 iterates over all feature indices i occuring in F. A fused measurements is constructed for each feature $\mathbf{y}_i$. Within this inner loop, subscripted state $\mathbf{x}_\alpha$ and mean $\boldsymbol{\mu}_\alpha$ vectors occur. These refer to the sub-state which the pixel measurements of the currently considered feature directly depend upon, i.e., the camera state $\mathbf{x}_v$ and the feature state $\mathbf{y}_i$. The loop body consists of the steps already outlined above.

First, the set of pixel measurement locations $\mathbf{U}^{\mathcal{C}}$ is obtained by rasterising the feature outline warped to the current image, cf. line 5. Individual pixel measurements are computed in lines 6–11, where the measurement model is linearized at $\boldsymbol{\mu}^k$ to compute the Jacobian. The only step that requires further explanation is line 10, where the intensity noise variance for an individual pixel is computed by the function GetNoiseCovariance. We will postpone details of the intensity variance model that is used here until Section 7.8.2.

Next, the fused measurement is computed in line 12 and brought into the appropriate form for the IEKF update calculation in lines 13–15. After the fused measurements have been computed for all features, they are stacked into the final joint measurement in line 17. The remainder of the algorithm is identical to the IEKF update discussed in Section 3.11: The joint measurement is used to compute the posterior estimate $\boldsymbol{\mu}^{k+1}$ for the present iteration. Then, the whole process is repeated until the posterior estimate converges.

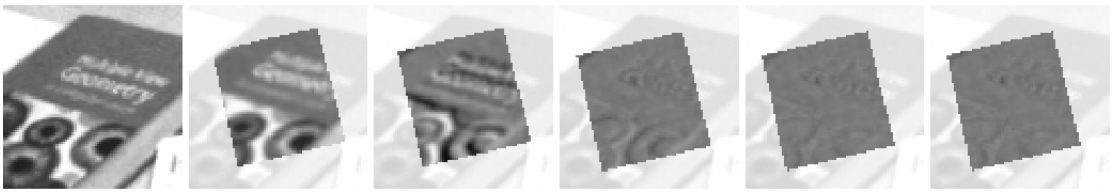
For the sake of brevity, Algorithm 17 has been presented in a monocular setting. For the stereo setup, the steps in lines 5–11 are repeated for the left image, as well. This simply contributes additional pixel measurements.

The computational overhead introduced by the iterative update is mainly caused by the re-computation and re-linearisation of the individual pixel measurements in every

Algorithm 17: IteratePlanarUpdate

Input: Prior belief distribution $\overline{\boldsymbol{\mu}}$, $\overline{\boldsymbol{\Sigma}}$, Current image C,
Features to be measured $F = \{i_1, \ldots, i_n\}$
Output: Posterior belief distribution $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$

$k := 0$
$\boldsymbol{\mu}^0 := \text{InitialUpdate}\left(\overline{\boldsymbol{\mu}}, \overline{\boldsymbol{\Sigma}}, C, F\right)$
repeat
  forall $i \in F$ **do**
    $\mathbf{U}^{\mathcal{C}} := \text{Rasterize}\left(w^{\mathcal{CA}}\left(L_i\,; \boldsymbol{\mu}^k\right)\right)$ `// yields` $\mathbf{U}^{\mathcal{C}} = \{\mathbf{u}_1^{\mathcal{C}}, \ldots, \mathbf{u}_m^{\mathcal{C}}\}$
    forall $\mathbf{u}_j^{\mathcal{C}} \in \mathbf{U}^{\mathcal{C}}$ **do**
      $h_j(\mathbf{x}_\alpha) := T_i\left(w^{\mathcal{CA}^{-1}}\left(\mathbf{u}_j^{\mathcal{C}}\,; \mathbf{x}_\alpha\right)\right)$
      $\mathsf{H}_j := \left.\dfrac{\partial h_j(\mathbf{x}_\alpha)}{\partial \mathbf{x}_\alpha}\right|_{\mathbf{x}_\alpha = \boldsymbol{\mu}_\alpha^k}$
      $\nu_j := C(\mathbf{u}_j^{\mathcal{C}}) - h_j(\boldsymbol{\mu}_\alpha^k)$
      $r_j := \text{GetNoiseCovariance}\,()$
    end
    $(\boldsymbol{\nu}_F, \mathsf{H}_F, \mathsf{R}_F) := \text{FuseMeasurements}\left(\left(\nu_1, \mathsf{H}_1, r_1\right), \ldots, \left(\nu_m, \mathsf{H}_m, r_m\right)\right)$
    $\boldsymbol{\nu}_{\mathbf{y}_i} := \boldsymbol{\nu}_F - \mathsf{H}_F\left(\overline{\boldsymbol{\mu}}_\alpha - \boldsymbol{\mu}_\alpha^k\right)$ `// accounts for model re-linearisation`
    $\mathsf{H}_{\mathbf{y}_i} := \begin{bmatrix}\mathsf{H}_F & \mathbf{0}\end{bmatrix}$ `// pad with 0 cols for` $\mathbf{x}_\beta$
    $\mathsf{R}_{\mathbf{y}_i} := \mathsf{R}_F$
  end
  $\mathsf{H} := \begin{bmatrix}\mathsf{H}_{\mathbf{y}_{i_1}} \\ \vdots \\ \mathsf{H}_{\mathbf{y}_{i_n}}\end{bmatrix}, \quad \boldsymbol{\nu} := \begin{pmatrix}\boldsymbol{\nu}_{\mathbf{y}_{i_1}} \\ \vdots \\ \boldsymbol{\nu}_{\mathbf{y}_{i_n}}\end{pmatrix}, \quad \mathsf{R} := \begin{bmatrix}\mathsf{R}_{\mathbf{y}_{i_1}} & & \\ & \ddots & \\ & & \mathsf{R}_{\mathbf{y}_{i_n}}\end{bmatrix}$
  $\mathsf{S} := \mathsf{H}\overline{\boldsymbol{\Sigma}}\mathsf{H}^\top + \mathsf{R}$
  $\mathsf{K} := \overline{\boldsymbol{\Sigma}}\mathsf{H}^\top\mathsf{S}^{-1}$
  $\boldsymbol{\mu}^{k+1} := \overline{\boldsymbol{\mu}} + \mathsf{K}\boldsymbol{\nu}$
  $k := k + 1$
until $|\boldsymbol{\mu}^k - \boldsymbol{\mu}^{k-1}| < \textit{threshold}$ **or** $k \geq \textit{max iterations}$
$\boldsymbol{\mu} := \boldsymbol{\mu}^k$
$\boldsymbol{\Sigma} := \overline{\boldsymbol{\Sigma}} - \mathsf{K}\mathsf{H}\overline{\boldsymbol{\Sigma}}$
return $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$



(a) (b) (c) (d) (e) (f)

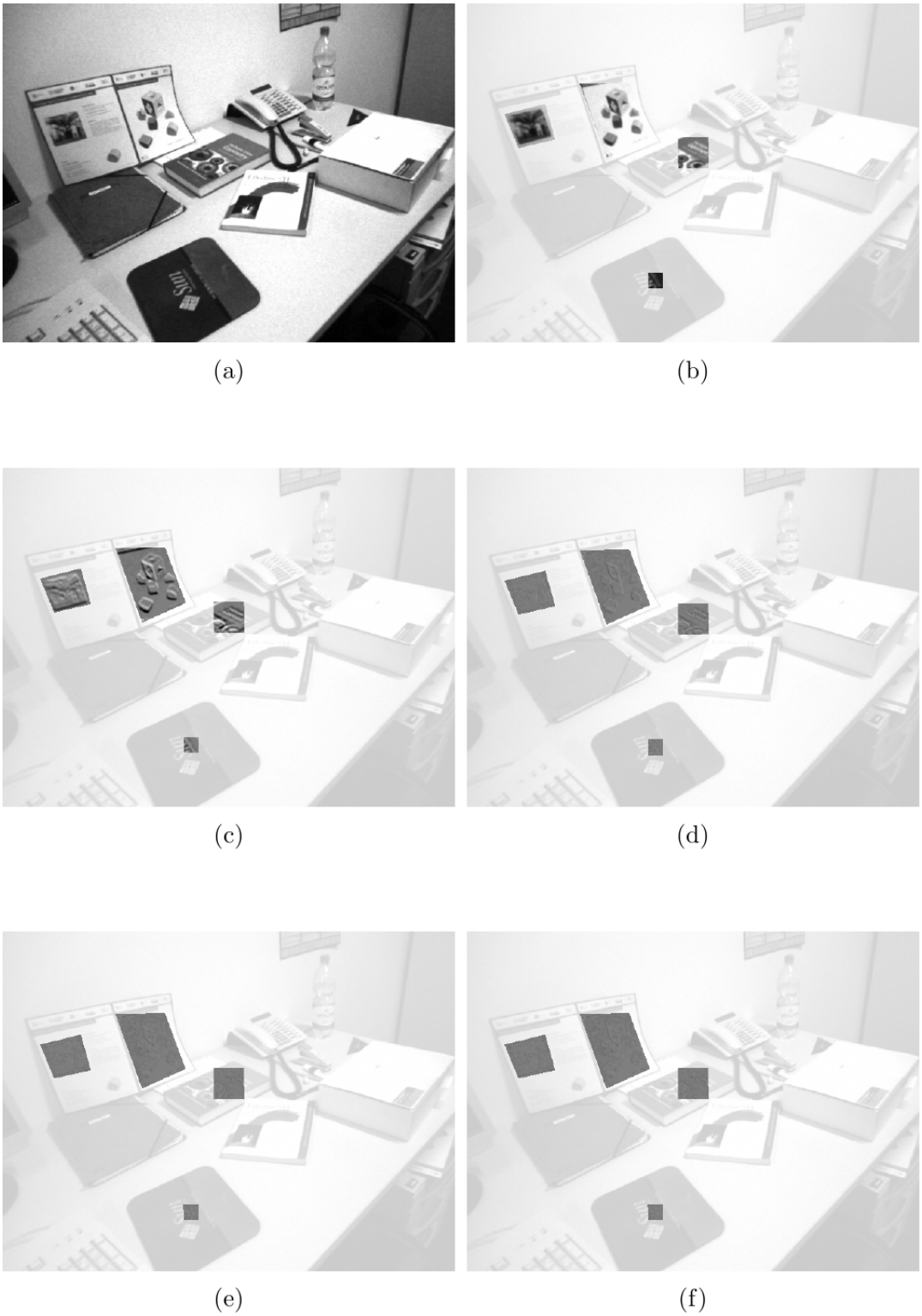
Figure 7.7.: Example of the measurement update iteration. (a) is the current image. (b) is the predicted feature template, warped according to the prior state estimate. (c)–(f) show the difference between the warped template and the current image. (c) is before the first iteration, i.e., the difference between (a) and (b). (d) is after the first iteration which is performed using the viewpoint-based measurement model. (e) and (f) are after the second and third iteration which are performed using the full planar measurement model.

iteration. The IEKF update itself is not much more expensive than a standard EKF update. As noted before, the posterior covariance matrix is not required in the loop and only computed after the mean has converged, which makes the loop iterations relatively cheap. Also, iterations can be restricted to the sub-state consisting of the camera pose and all features that are currently measured. The full state only has to be updated once, after the mean estimate has converged.

The iterative update is illustrated in Figures 7.7 and 7.8. Figure 7.7 shows the update for a single feature, while Figure 7.8 shows the process for several features that are updated simultaneously. In both figures, (a) shows the current image and (b) shows the predicted templates that are used for ZSSD search. The remaining subfigures (c)–(f) illustrate the difference between the predicted intensities and the current image at various points in the iteration. An intensity difference of zero is visualised as medium grey. Lighter and darker greys indicate positive and negative differences. The sum of squared intensity differences is minimised in the update step, taking into account prior information. Subfigure (c) respectively shows the initial difference, i.e., before the first iteration of the IEKF. Much of the differences disappear in (d) after the first iteration with the restricted model. Subfigures (e) and (f) show the situation after the second and third iteration which are performed using the direct intensity measurement model. It can be seen that the first iteration brings the state to a good initial solution, which is then further refined using the direct model. Typically, convergence occurs after very few iterations. In the examples, the estimate has converged after the third iteration (f).

7.7. Initialisation of New Planar Features

In this section we discuss the initialisation of a planar feature into the probabilistic map. The process is similar to the initialisation of a view-point based feature. The difference here is, that additionally we have to initialise the feature plane normal. This is done in a two-step process. First, the normal is initialised as pointing towards the reference pose,



(a) (b)

(c) (d)

(e) (f)

Figure 7.8.: Measurement update iteration for multiple features. (a) current image. (b) predicted feature templates. (c) before the first iteration. (d) after the first, viewpoint-based, iteration. (e), (f) after the second and third iteration.

i.e., the right eye of the camera. The uncertainty of the normal is set to a large value. In the second step, the pixel intensities of the left image are used in an update step. During this update, the normal and the inverse depth are adjusted.

7.7.1. Initialisation with Camera-Facing Normal

Let us take a closer look at the first step. Because of the similarity to view-point based features, we will keep the discussion rather brief. For additional details we refer to Section 6.5.2 where view-point based initialisation is introduced.

Suppose that we want to initialise a planar feature in the current frame. As input to the initialisation we have

- the polygonal outline of the feature, L,
- the image coordinate of the feature center, $\mathbf{u}^{\mathcal{I}}$, and
- a measurement $\mathbf{z}_d$ of the disparity at $\mathbf{u}^{\mathcal{I}}$.

The point $\mathbf{u}^{\mathcal{I}}$ is an arbitrary fixed point within the feature outline. Although we refer to the point $\mathbf{u}^{\mathcal{I}}$ as the "feature center" it does not need to lie exactly at the center of the planar patch.

To initialise the planar feature we have to initialise the feature state $\mathbf{y}$ into the probabilistic state, and we have to fix the static components of the feature model. The static components of the model are the feature ray $\mathbf{m}^{\mathcal{A}}$, the template image T, and the feature outline L. The feature ray $\mathbf{m}^{\mathcal{A}}$ is chosen as the ray through the feature center $\mathbf{u}^{\mathcal{I}}$,

$$\mathbf{m}^{\mathcal{A}} = \pi^{+}\left(\mathbf{u}^{\mathcal{I}}\right), \tag{7.71}$$

where π^{+} is defined as back-projecting to the plane $z = 1$ and scaling to unit length, cf. Equation 6.28. The template image T is a copy of the current right image. The feature outline L is provided as an input to the initialisation.

To augment the new feature vector $\mathbf{y}$ into the state we need an inverse measurement model which computes the new feature vector as a function of the measurement and the current state (cf. Section 4.8.2). The feature vector consists of the anchor pose $\mathbf{c}$, $\boldsymbol{\phi}$, the inverse depth ρ, and the feature normal $\boldsymbol{\theta}$. The initialisation of the first three is identical to the view-point based model. The normal vector $\boldsymbol{\theta}$ is not observable from the reference image, therefore in a first step, it is initialised with high uncertainty as pointing directly towards the anchor. In a second step, described below, it will be updated with measurements from the left image.

The camera-facing normal is computed as

$$\mathbf{z}_{\boldsymbol{\theta}} = \mathbf{n}^{-1}\left(-\mathbf{m}^{\mathcal{A}}\right) \tag{7.72}$$

where

$$\mathbf{n}^{-1}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \arctan\frac{x}{z} \\ \arctan\frac{y}{\sqrt{x^2+z^2}} \end{pmatrix} \tag{7.73}$$

converts a vector to polar azimuth and elevation angles. For the initialisation, we model $\mathbf{z}_{\boldsymbol{\theta}}$ as an artificial noisy measurement of the actual normal parameters $\boldsymbol{\theta}$,

$$\mathbf{z}_{\boldsymbol{\theta}} = \boldsymbol{\theta} + \boldsymbol{\delta}_{\boldsymbol{\theta}}. \tag{7.74}$$

The noise term $\boldsymbol{\delta}_{\boldsymbol{\theta}}$ is assumed zero-mean Gaussian with variance $\mathsf{R}_{\boldsymbol{\theta}}$. In this way, we can model our belief about the true normal parameters as

$$p(\boldsymbol{\theta} \,|\, \mathbf{z}_{\boldsymbol{\theta}}) = \mathcal{N}(\mathbf{z}_{\boldsymbol{\theta}}, \mathsf{R}_{\boldsymbol{\theta}}). \tag{7.75}$$

The variance $\mathsf{R}_{\boldsymbol{\theta}}$ is chosen to reflect the high uncertainty in the "measurement". In our implementation we set it to a diagonal matrix that corresponds to a standard deviation of 45° in every direction.

Similar to view-point based initialisation, the disparity measurement $\mathbf{z}_d$ is modeled as a noisy observation of the hypothetical perfect measurement d, that is, the actual disparity of the point where the ray through $\mathbf{u}^{\mathcal{I}}$ hits the scene.

$$\mathbf{z}_d = d + \delta_d \tag{7.76}$$

The noise term δ_d is assumed zero-mean Gaussian with variance σ_d^2. The variance describes the accuracy of the disparity estimation method.

Now we can define the inverse measurement model as a function of the current state and the ideal, noise-free measurements of disparity d and polar normal $\boldsymbol{\theta}$.

$$\mathbf{y} = \begin{pmatrix} \mathbf{c} \\ \boldsymbol{\phi} \\ \rho \\ \boldsymbol{\theta} \end{pmatrix} = \mathbf{g}(d, \boldsymbol{\theta}, \mathbf{x}) = \begin{pmatrix} \mathbf{r}^{\mathcal{WC}} \\ \log\left(\mathbf{q}^{\mathcal{WC}}\right) \\ d\frac{z_m}{f_x b} \\ \boldsymbol{\theta} \end{pmatrix}. \tag{7.77}$$

The anchor pose, $\mathbf{c}$, $\boldsymbol{\phi}$, is obtained as a copy of the current camera pose. The inverse depth ρ is computed from the disparity d at the feature center. For a detailed derivation we refer to Section 6.5.2. The polar normal parameters are initialised to the given assumed value $\boldsymbol{\theta}$.

The Jacobians $\frac{\partial \mathbf{g}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{g}}{\partial d}$ are the same as for the view-point based model, cf. Appendix A.5. The Jacobian $\frac{\partial \mathbf{g}}{\partial \boldsymbol{\theta}}$ is the identity matrix.

7.7.2. Update with Left-Eye Measurements

Because we are using a stereo camera, we can immediately use intensity measurements to refine the initial feature estimate. We perform an iterative update step for the new feature, as described in Section 7.6. However, only pixels from the left image are used to form the measurement vector. The update will only influence the inverse depth ρ and the normal vector $\boldsymbol{\theta}$ parameters of the new features. Neither the current camera pose nor the anchor parameters $\mathbf{c}$, $\boldsymbol{\phi}$ will be updated. This is because the measurement is made from the same camera pose from which the feature was initialised. The current pose and the anchor pose are initially perfectly correlated.

This second step of the initialisation procedure is exemplified in Figure 7.9. A feature is initialised on the book in the center of the image. In the top row of Figure 7.9, the situation is shown after the feature has been initialised into the state, but before the update with the left image measurements has occurred. The feature outline and normal vector are overlaid on the left and right image, respectively. The second and third row show the estimates after the 4th and 8th iteration of the left image update. Note, that the projected outline does not change in the right image. This is because the current right image is the template image for the feature and the outline is defined with respect to this image. It can be seen that the normal converges to a visually correct direction.

Further details are shown in Figure 7.10. The figure shows the predicted left-eye intensities and the difference to the observed image before the update and after the 2nd through 8th iteration. Clearly, the image difference is reduced in every iteration. In this example, the IEKF converges after the 8th iteration.

7.8. Remarks

In the following, we discuss some further details of our implementation of the planar feature model. In Section 7.8.1, we discuss how to speed up fused measurement computation by reordering matrix multiplications. In Section 7.8.2, we describe how per-pixel noise is modeled in our implementation. Finally, in Section 7.8.3, we propose an image preprocessing method to address sampling artefacts caused by scale differences in the template and current images.

7.8.1. Jacobian Decomposition for Efficient Fused Measurement Computation

We have shown in Section 7.4.3 that the Jacobians of the pixel measurement functions can be split into a 1×9 factor $\mathsf{H}_{T;j}$, which is different for every pixel, and a 9×10 factor H_w, which is constant for all pixels. We can use this to speed up the summation over the pixel measurements that is performed in measurement fusion.

Consider the sums in line 1 and 2 of Algorithm 15. For our pixel measurements we have

$$\mathsf{H}_j = \mathsf{H}_{T;j}\mathsf{H}_w. \tag{7.78}$$

where H_w is constant over all j. We can rearrange the sum in line 1 of Algorithm 15 as follows

$$\Delta\boldsymbol{\Omega} = \sum_{j=1}^{m} \mathsf{H}_j^\top r_j^{-1} \mathsf{H}_j \tag{7.79}$$

$$= \sum_{j=1}^{m} \mathsf{H}_w^\top \mathsf{H}_{T;j}^\top r_j^{-1} \mathsf{H}_{T;j} \mathsf{H}_w \tag{7.80}$$

$$= \mathsf{H}_w^\top \left(\sum_{j=1}^{m} \mathsf{H}_{T;j}^\top r_j^{-1} \mathsf{H}_{T;j} \right) \mathsf{H}_w. \tag{7.81}$$

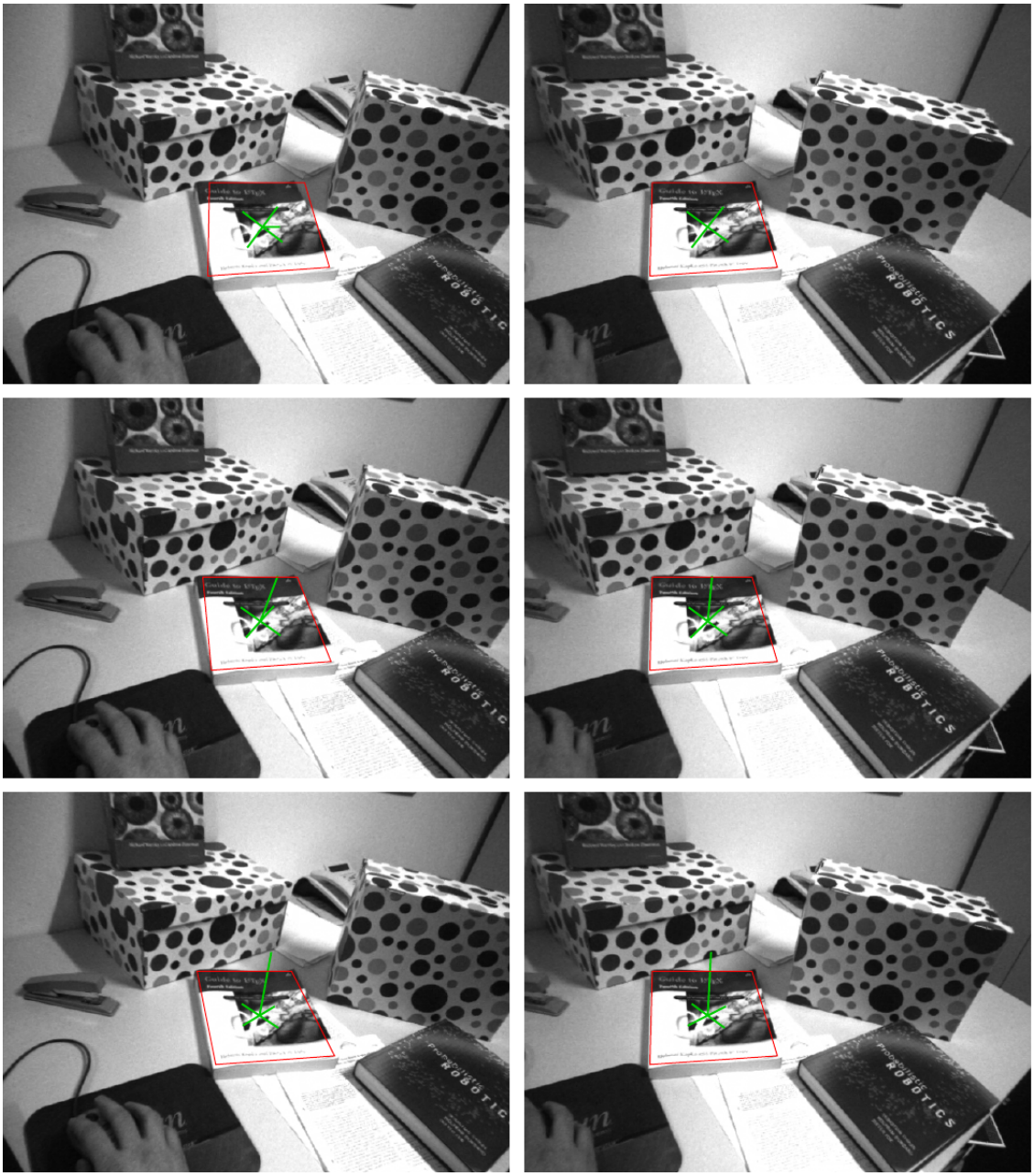


Figure 7.9.: Updating inverse depth and normal with measurements from the left eye. In the left column, the current left image of the camera is shown. In the right column, the current right image, i.e., the template image, is shown. The estimated feature outline and normal vector are overlaid. The first row shows the initial estimate where the normal vector is pointing towards the right camera center. The second and third row show the corrected estimates after the 4th and 8th iteration, respectively.

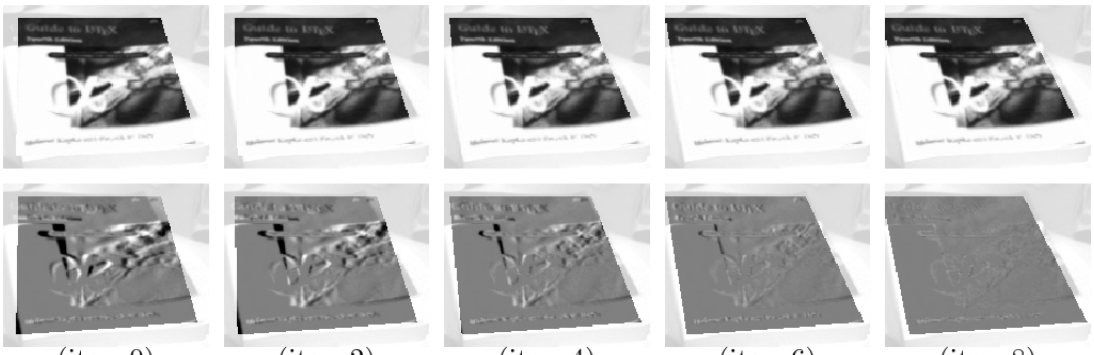


Figure 7.10.: Updating inverse depth and normal with measurements from the left eye. More details are shown for the example given in Figure 7.9: Every column shows the predicted intensities for the left eye (top) and the difference to the observed intensities (bottom) for one iteration of IEKF. The iteration number is indicated below each column.

Similarly, we can rearrange the sum in line 2 as

$$\Delta\boldsymbol{\xi}^{+} = \sum_{j=1}^{m} \mathsf{H}_j^{\top} r_j^{-1} \nu_j \tag{7.82}$$

$$= \sum_{j=1}^{m} \mathsf{H}_w^{\top} \mathsf{H}_{T;j}^{\top} r_j^{-1} \nu_j \tag{7.83}$$

$$= \mathsf{H}_w^{\top} \left(\sum_{j=1}^{m} \mathsf{H}_{T;j}^{\top} r_j^{-1} \nu_j \right). \tag{7.84}$$

By pulling H_w out of the summation we can avoid the matrix multiplication (7.78) for every pixel. Computing $\Delta\boldsymbol{\Omega}$, $\Delta\boldsymbol{\xi}^{+}$ in this way requires an order of magnitude less operations. Computing (7.78) for m pixels and evaluating the original sums requires $680m$ floating point operations. In contrast, the evaluation of the modified expressions above requires only $(54m + 4760)$ floating point operations.

7.8.2. Measurement Noise Variance

The planar feature measurement model (7.10) subsumes unmodeled variations of the measured intensities into additive noise terms δ_j. We model $\delta_j \sim \mathcal{N}(0, r_j)$ as zero-mean Gaussian with variance r_j, and assume it is uncorrelated between pixel locations. An issue that we have not addressed so far is how to choose the measurement noise variance.

Various influences contribute to the noise term, such as sensor noise, illumination changes, aliasing, and linearisation effects. Molton et al. (2003) list and model major sources of uncertainty during image alignment. The various sources contribute noise terms that are constant per pixel, depending on the pixels gradient, and depending on

the warp parameters, respectively. Experiments in their work indicate that a simplified model with only a constant variance term is sufficient. Using a more detailed uncertainty model gave very similar results. They choose $\sigma_1 = 10$ intensity levels standard deviation for the constant noise term w_1. We adopt this value here.

In experiments with real footage from the Bumblebee® stereo camera we have found that incorporating an additional gradient-dependent term improves tracking stability for our setup. This term models intensity differences caused by small pixel offsets $\mathbf{w}_2$, which are represented as 2D isotropic Gaussian distributed. As explained by Molton et al. (2003), this models aliasing effects as well as warp linearisation errors. We set the standard deviation of the pixel offset to $\sigma_2 = 1$ in every direction.

In summary, we model the intensity error as

$$\delta_j = w_1 + \nabla T(\mathbf{u}_j^{\mathcal{A}})\mathbf{w}_2 \tag{7.85}$$

which gives the measurement noise variance

$$r_j = \sigma_1^2 + \nabla T(\mathbf{u}_j^{\mathcal{A}})\ \sigma_2^2\ \nabla T(\mathbf{u}_j^{\mathcal{A}})^\top \tag{7.86}$$

at the jth pixel.

7.8.3. Anisotropic Smoothing for Scale Adjustment

The pixels we observe in both the template and the current image are sampled from underlying continuous projections of the real world. In general, the scales of both images will be different, i.e., the world region generating a pixel in the template and the corresponding pixel in the current image will be different.

Consider a situation where the camera pose for the current image is much closer to the feature plane than the camera pose for the template image. In this case, the current image shows small-scale details which are not visible template image. The effective resolution of the current image is larger. It seems reasonable in such a situation to suppress the small-scale details in the current image by smoothing the current image. That is, the scale of the current image is adapted to the scale of the template image. Similar considerations can be made when the current camera is further away from the feature or is observing it from a different angle.

We have experimented with adapting the scale of the current and template image using anisotropic Gaussian smoothing of both images. It turns out that this preprocessing step can lead to minor improvements in estimation accuracy. The smoothing step can be rather computationally expensive, though. The benefits are largest if there is a severe view-point difference between the current and template image. In this case, the computation cost is also the highest, because large Gaussian smoothing masks must be employed to handle the resulting severe image distortion. Consequently, this preprocessing step is optional in our final implementation. We briefly sketch the approach in the following.

The basic idea is illustrated in Figure 7.11. The template image T and the current image C are both generated by sampling a projection of the fronto-parallel image P of the feature. P is the ground truth continuous texture of the feature plane in the world. The coordinate frames of the template image and the current image are each related by a homography

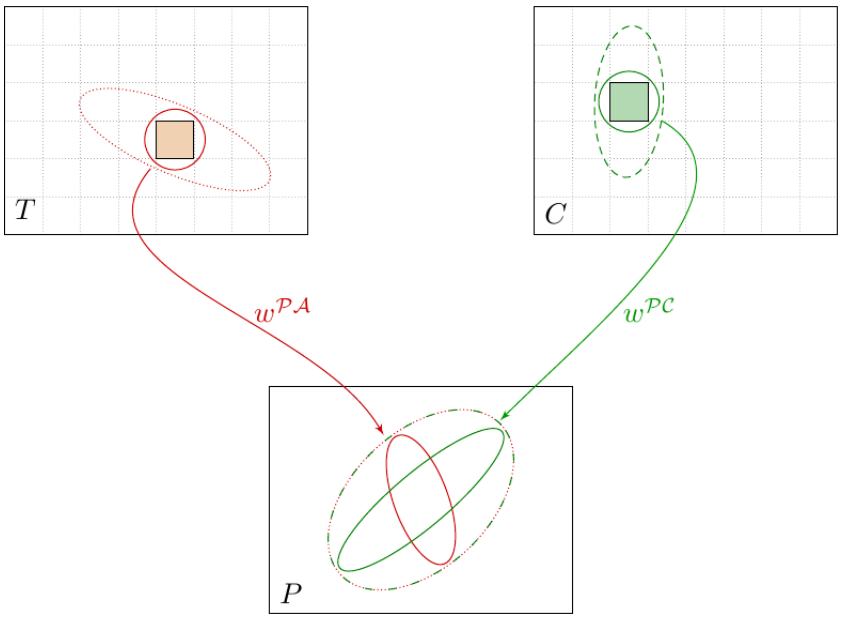


Figure 7.11.: Adjusting the scale of the template and the current image by anisotropic Gaussian smoothing. Corresponding pixels in the template image T and the current image C have different support regions when transformed to the feature plane P. By convolving both images with appropriate Gaussians, the resulting support regions can be brought in alignment, as illustrated by the dotted and dashed ellipses, respectively.

transformation to the coordinate system of P. In Figure 7.11, these transformations are denoted as $w^{\mathcal{PA}}$ and $w^{\mathcal{PC}}$, respectively. Locally, the homographies can be approximated as affinities. With this approximation, a Gaussian in one frame is projected to a Gaussian in the others.

A pixel in T (the red square in Figure 7.11) can be modeled as being generated by convolution of the underlying continuous projection with a Gaussian (illustrated by the red circle). Likewise, the corresponding pixel in C (green square) is modeled as being generated by convolution with a Gaussian (green circle).

When projected to P, the isotropic Gaussians are transformed to anisotropic Gaussians. In general, the support regions for the red and green pixel in P do not match (as illustrated by the red and green ellipses in P). The goal is to convolve T and C with appropriate anisotropic Gaussian kernels, such that the resulting support regions project to the same region in P. This is illustrated by the dotted and dashed ellipses, respectively. The dotted red ellipse in T indicates the effective support region for the red pixel after smoothing image T. The dashed green ellipse in C indicates the effective support region for the green pixel after smoothing image C. The smoothing kernels are chosen such that the support regions match when projected to P. The parameters of appropriate smoothing kernels can be computed from the homography $w^{\mathcal{CA}}$ between the current and template image.

Figure 7.12 shows the application of the technique to an image from a real sequence. It can be clearly seen that the unprocessed current image (d) and the image predicted from the unprocessed template (g) differ in effective resolution. The current image contains more details in the horizontal direction. The image predicted from the template contains more details in the vertical direction. When the images are preprocessed with appropriate Gaussian smoothing the differences are corrected, cf. (e) and (h).

7.9. Experimental Evaluation

The goal of this section is to evaluate the performance of planar features in the context of a visual SLAM system. We will use rendered image sequences to directly compare measurement and reconstruction accuracy of planar and point features. Moreover, we assess the applicability of planar features on real image sequences.

We want to verify the following expectations. First of all, full visual SLAM operation should be possible using only planar features. This should be verified both on real and simulated image sequences.

Second, the direct intensity measurement method inherently provides subpixel accuracy for the planar feature model. Moreover, the feature normal vector is estimated as part of the probabilistic state representation. Thus, mapping and localisation accuracy should be higher than for standard point feature matching. It should rival the accuracy that can be achieved with point features using subpixel-accurate matching of accurately predicted appearance templates using ground-truth normal vectors.

Finally, planar measurements capture the homography transformation between feature template and current image. This provides more information than the 2D image position of a point feature measurement. In theory, a single planar feature should be enough to fully constrain the camera motion.

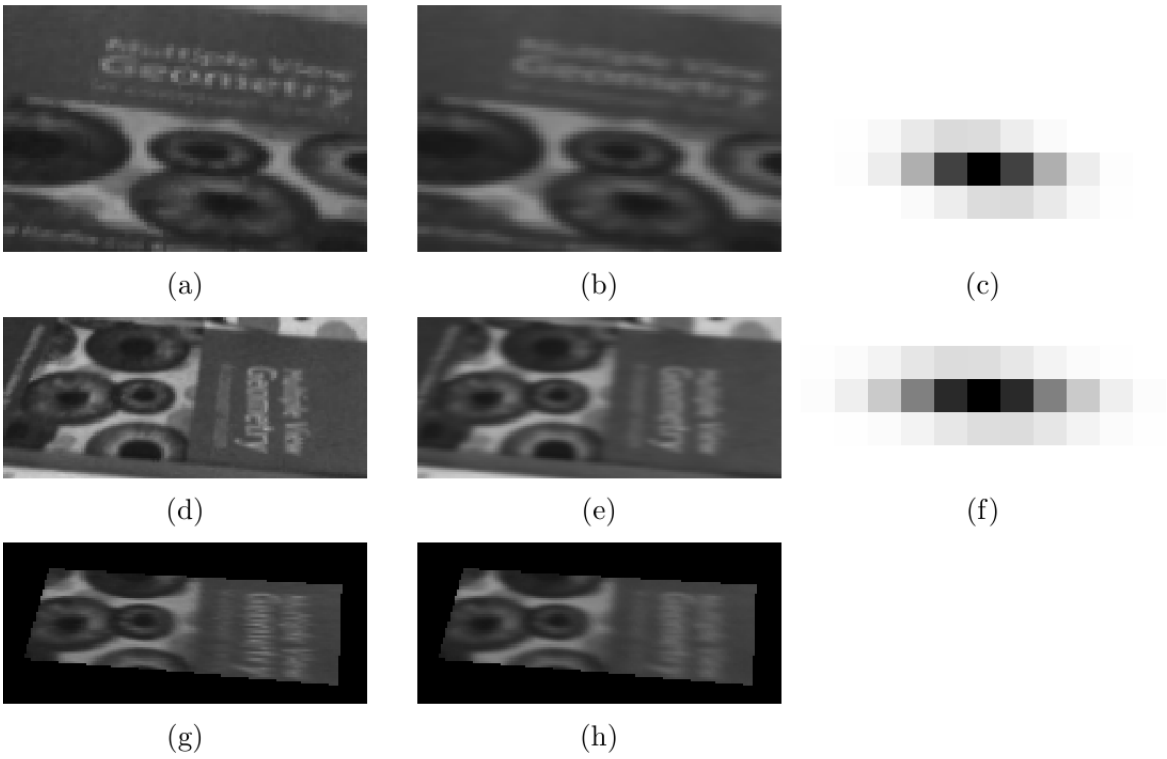


Figure 7.12.: Adjusting the scale of the current and template image by anisotropic Gaussian smoothing. (a) shows the template image, (b) shows the template image smoothed with the kernel (c). (d) shows the current image, (e) shows the current image smoothed with the kernel (f). (g) shows the predicted image intensities obtained by warping the raw template image (a). (h) shows the predicted image intensities obtained by warping the smoothed template image (b). It can be seen, that (d) has more details than (g) in the horizontal direction, while (g) has more details than (d) in the vertical direction. In the smoothed images (e) and (h) these differences in effective resolution have been corrected.

In Section 7.9.1 we discuss details of the SLAM system with planar features that is employed for the experiments. In Section 7.9.2 we compare planar feature and point features using rendered image sequences with known ground truth. In Section 7.9.3 we test visual SLAM with planar features on real image sequences.

7.9.1. Planar Features SLAM System

The application of planar features in a full visual SLAM system requires further details not discussed yet in this chapter. In particular, this concerns two issues. First, a full system requires an approach for the automatic detection and selection of new planar feature candidates. Second, it needs to be able to robustly handle measurement errors. The planar feature SLAM system used in the experiments is still a prototypical implementation where these issues are handled as follows.

Planar features in our system are initialised by selecting new feature candidates manually. When a new feature should be initialised, the system is provided with the feature outline in the reference image and a rough estimate of the disparity at the feature center. A list of outlines and disparities for all features used in an experiment must be provided by the user. This is handled in the following way. A point-based SLAM system is run on the experimental sequence. During this run, if a planar feature is to be initialised in the current image, the user may select with the mouse a polygonal outline in the reference image. It is the user's responsibility to ensure that the selected outline corresponds to a planar scene surface. The disparity estimate is computed from the depth of the point feature that is closest to the centroid of the polygon (in the reference image). All planar feature candidates collected in this manner are stored in a list, which is later used by the planar SLAM system to initialise features. Given an image outline and disparity estimate, feature initialisation proceeds as described in Section 7.7. First, the feature is initialised at the approximate depth with a camera-facing normal. Then, measurements in the left image are used to update the normal and inverse depth estimates.

In our experiments with real sequences, measurement errors occur chiefly due to occlusion or partial occlusion of features. Only very basic error detection is currently implemented: During the initial template search stage of the measurement process, the best match to the predicted planar template is searched. We compute the average intensity difference per pixel between the best match and the planar template. If the average difference is above a threshold, the measurement is discarded. This approach reliably rejects fully occluded features. However, partial occlusions are often not detected when only a small part of the feature is occluded. In this case, the influence on the average intensity difference is small.

For a practical system, robust solutions for the above mentioned issues would be required. With respect to automatic planar structure detection, we could resort to several plane detection methods that have been proposed in the literature. In particular the hypothesis testing approach of Martínez-Carranza & Calway (2009a) is a promising candidate.

To increase robustness, a method should be developed to predict or detect partial occlusions of a feature. Pixels from the visible portion of the feature could still be used as valid measurements. Furthermore, it would be desirable to extend the Joint Compatibility

approach to planar features. This is challenging because the approach must be integrated with the iterative update procedure.

For the purpose of experimental evaluation, feature initialisation and error detection were handled pragmatically as described above. Moreover, because partial occlusion is currently a failure mode for our system, we avoided them in the design of the experiments.

7.9.2. Comparison with Point Features on Simulated Sequences

Evaluation Procedure

We compare the accuracy of the planar feature model and a point feature model on rendered image sequences. We use two scenarios. The first scenario compares the accuracy of planar and point features on a minimal example. The map consists of a single planar feature or three point features, respectively. The accuracy of the estimated camera trajectory is compared for both setups. We analyse the influence of various point feature measurement methods, as well as motion-blur artefacts and additive noise in the sequence.

In the second scenario we use multiple features and a more realistic sequence. The scene consists of several planar segments. The trajectory contains several loop closures. The influence of motion-blur and additive noise is analysed.

We compare two feature models:

- **"planar features"** This is the planar feature model using direct intensity measurements that was described in this chapter.
- **"point features"** This is the inverse depth bundle model described in Chapter 6. Measurements are carried out using standard pixel-accurate ZSSD template matching. We further examine the effect of subpixel-accurate measurement refinement, as well as *a priori* known normal directions.

For the experiments, both models were implemented within full systems based on the visual SLAM framework described in Chapter 4. As detailed above, the planar system deviates from the framework in that it lacks automatic initialisation and robust data association.

The test sequences were generated and evaluated using the SLAMDUNK framework. For each run, the camera estimate of the visual SLAM system is initialised to the ground-truth pose with no uncertainty. The performance of both models on the test sequences is evaluated according to the following measures.

- **Absolute camera position error.** The absolute camera position error is defined as the Euclidean distance between the ground truth and the estimated camera pose. It is computed for each frame of the sequence and plotted over time.
- **Individual camera pose component errors.** We analyse the deviation of individual components of the camera pose from their ground truth values. These comprise the x, y, z translations and the components of the quaternion orientation. These errors are computed for each frame of the sequence and plotted over time. The plots also include the 3σ bounds of the estimated uncertainty which allows to assess the consistency of the estimate.

- **Best-fit map error.** We define the map error as the root mean squared distance between estimated map features and corresponding ground truth features. We compute a best-fit rigid transformation of the estimated map such that the map error is minimized. The best-fit map error is defined as the residual map error after applying this best-fit transformation. It is computed for each frame of the sequence and plotted over time.

- **Per feature absolute measurement error.** For a given feature, the perfect 2D measurement is the ground truth stereo projection of the feature center into the ground truth reference camera. The absolute measurement error is defined as the distance (in pixels) between the actual feature measurement and the perfect measurement. It is computed for each frame of the sequence and plotted over time.

- **Per feature measurement error distribution.** For a given feature, we plot the 2D distribution of their measurement error in the reference image over the sequence. Here, the measurement error is defined as the 2D offset (in pixels) between the actual feature measurement and the perfect measurement.

The comparison of measurement errors requires further explanation. It is difficult to directly compare measurement errors for point and planar features. Remember that planar feature measurements are high-dimensional intensity vectors while point feature measurements are 2D image positions. To compare both feature types, we compute artificial 2D "measurements" for the planar features. This is done by computing the 2D image position of the planar feature center using the posterior state estimate. As with point features, the perfect measurement is the ground truth projection of the feature center. For the single feature setup of the first scenario, the posterior state is a result of measurements of this one feature only. Thus, the above procedure provides a fair estimate of the measurement accuracy. For the second scenario, we will not evaluate the measurement error because multiple planar features are used. In this case, the posterior state is calculated using the joint information from all features. The estimate of one feature is thus improved by measurements of other features. This would result in overly accurate artificial measurements.

A Single Planar Feature

In the first set of experiments we examine the local properties of the planar feature model on a minimal example. A simple scene consisting of a single planar object is employed. We use the minimal number of features that is required to fully constrain the camera motion. For this, three point features are needed. For the planar model, a single, sufficiently large feature should be enough.

In particular, the purpose of this first experiment is to analyse the following issues:

- Is a single feature sufficient to obtain scene and camera pose estimates in practice?

- Can we observe increased measurement accuracy with respect to standard point feature matching?

- Direct intensity measurement should provide sub-pixel accurate measurements. How does the planar model compare to subpixel-accurate point measurements?

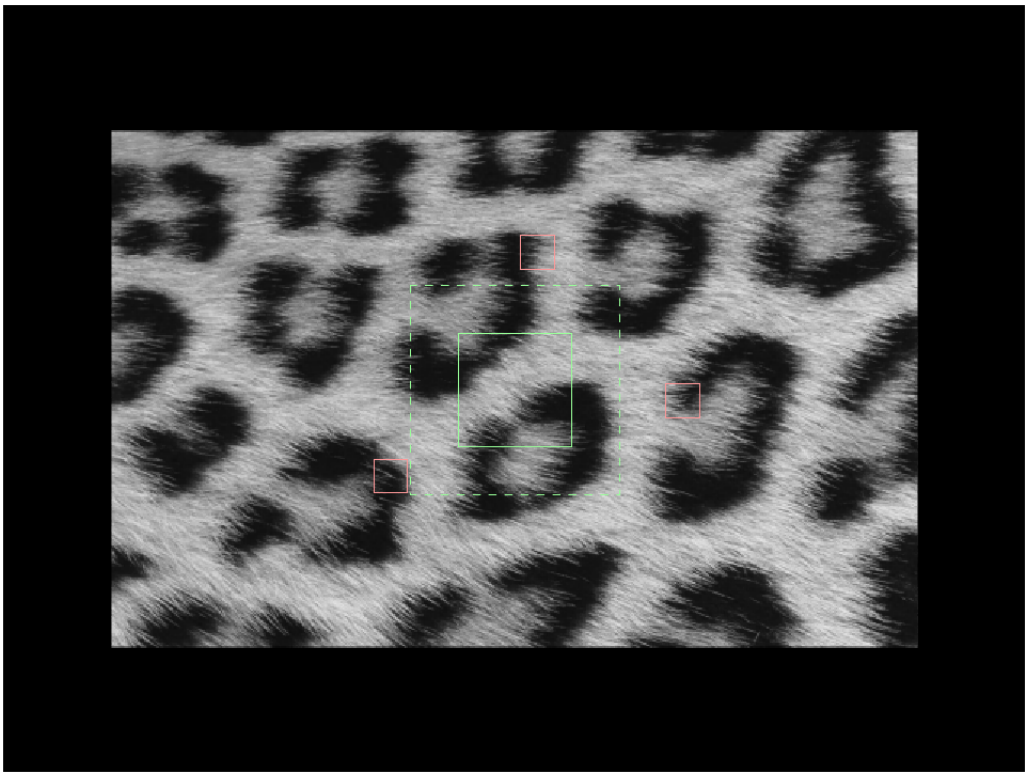


Figure 7.13.: The first image of the rendered "fur" sequence. Three 21×21 point features are initialised as shown in red. One 71×71 planar feature is initialised as shown in green. The dotted green outline indicates a 131×131 planar feature (which will be used in a later experiment).

- The highest quality point feature result could be achieved if the feature normals were exactly known, because this improves the predicted appearance of the matching template. In the planar model, the feature normal is an estimated parameter. How does the planar feature compare to point features with *a priori* known normals?

To answer these questions, we use the following simple setup.

"Fur" Sequence. The "fur" scene comprises a single well-textured plane. The camera is modeled after a Point Grey Bumblebee® stereo camera with 640×480 resolution and 65° horizontal field of view. The camera moves along a smooth, hand-crafted trajectory, during which the plane is viewed from varying distance (0.4 to 1.7 m) and varying angles (0° to 70°). Initially, the camera is 1 m away from the plane, and orthogonally facing it. The sequence comprises 500 frames (about 16 seconds). Figures 7.13 and 7.14 show selected images of the rendered sequence.

Comparison to Point Features. The sequence is processed with the point-based and the planar SLAM system. In both cases, the camera is initialised in the correct pose

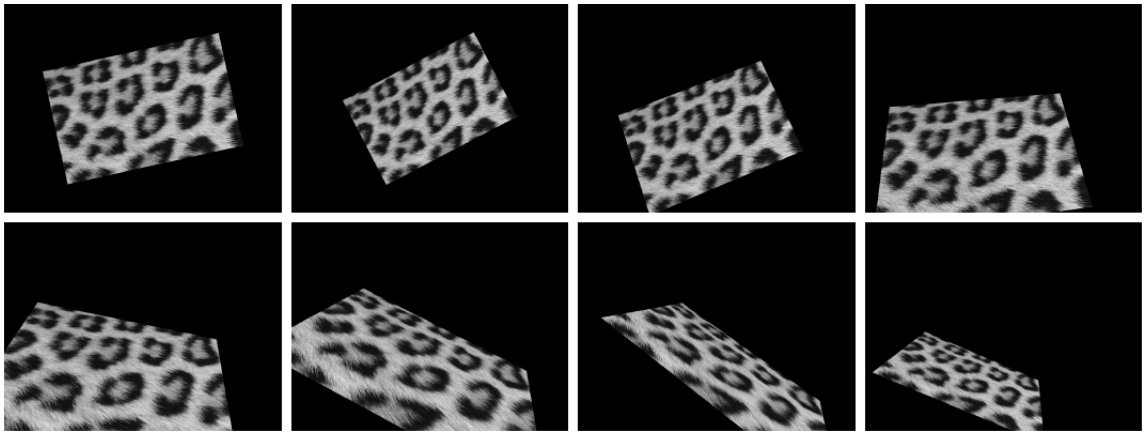


Figure 7.14.: Sample images from the rendered "fur" sequence. First row: 2, 4, 6, 8 seconds into the sequence. Second row: 10, 12, 14, 16 seconds into the sequence.

with no uncertainty. One planar and three point features, respectively, are initialised as follows. A single planar quadratic planar feature of 71×71 pixels is initialised in the center of the image. Three inverse-depth bundle point features of 21×21 pixels are initialised on maxima of the FAST feature detector. The image is exhaustively searched for FAST maxima. Among the detected maxima, feature locations are selected by hand such that they span a triangle in the image center. The feature locations are illustrated in Figure 7.13. The image area spanned by the point features is slightly larger than that covered by the planar feature, thus slightly better constraining the pose estimate. The overall number of pixels covered is higher for the planar feature though, which should counteract this effect.

The experiment has been carried out with different settings of motion blur and additive noise. For motion blur the shutter time is set to 0 ms (no blur), 33 ms, and 99 ms. Each of these setting is combined with additive Gaussian noise with standard deviation $\sigma \in \{0, 10, 20, 30\}$ intensity levels. In the following, we present in detail results for a setup with moderate image degradation, i.e., 33 ms motion blur and $\sigma = 10$ intensity noise. The results are qualitatively similar for the other experiments. As expected, we see a loss of accuracy in both the planar and point feature models with increasing motion blur or additive noise.

The sequence is tracked successfully by both models. A single planar feature is sufficient for this purpose as we anticipated. It can adequately replace the three point features in the experiment.

A comparison of the absolute camera position errors is shown in Figure 7.15. The red plots represent the setup with three point features. The blue plots represent the setup with a single planar feature. In the beginning of the sequence (approximately up to frame 100) the camera moves backwards away from the planar object. The triangulation baseline in the images gets smaller. Localisation error and jitter increase for both setups. Subsequently, the camera moves closer to the plane and errors decrease again. Towards the end, the plane is viewed in a steep angle. This causes increased error especially for

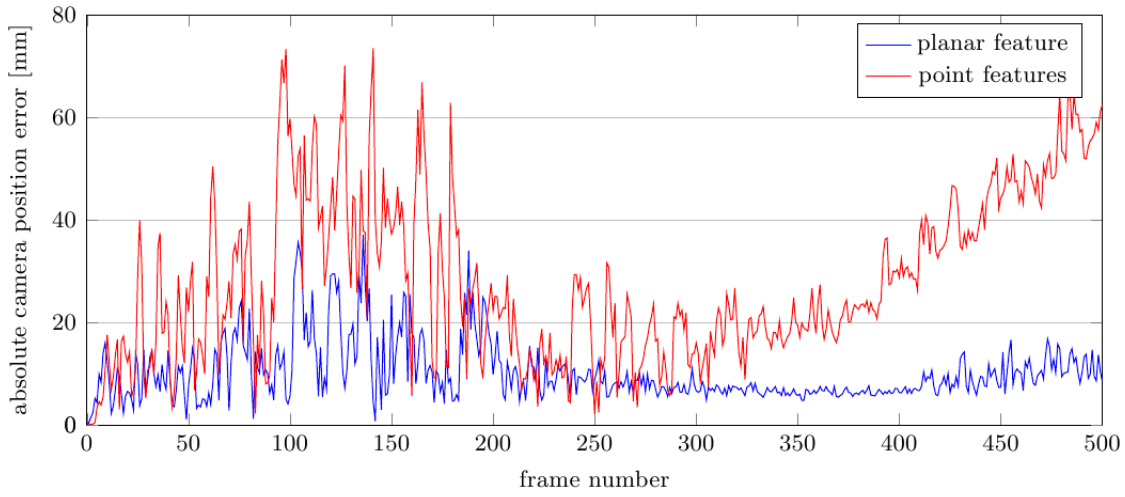


Figure 7.15.: Absolute camera position error for the "fur" sequence with 33 ms motion blur and $\sigma = 10$ intensity noise. The plot compares a single 71×71 planar feature and three 21×21 point features.

the point feature model. For the planar model, the error mostly stays around 10 mm. The error for the point feature model is three to four times larger towards the end of the sequence. Clearly, the trajectory obtained by the planar feature is more accurate and smooth.

Figure 7.16 gives a closer look at the individual components of the camera state estimate. The plots shows the absolute errors in the x, y, z components of the camera translation $\mathbf{r}$ and quaternion rotation $\mathbf{q}$. For the unit-less quaternion components, an error of $\mathbf{q}_x = 0.05$ corresponds to $\approx 6°$ rotation about the x axis. The 3σ bounds of the estimated uncertainty are illustrated by the dashed curves, respectively. The plots confirm the result of the absolute camera position error. The planar feature has smaller absolute error and the plots show less jitter.

However, the analysis of the uncertainty bounds points to a potential problem with the planar feature model. The pose estimates become over-confident. Over-confidence is a well-known and unavoidable problem in EKF based SLAM approaches. However, it seems that for the planar model it occurs earlier than for the point model. This is in particular visible in the q_z component in the lower left plot in Figure 7.16. Both models show systematically biased errors towards the end of the sequence. The planar error is above the 3σ bound for the second half of the sequence. The point feature error is larger but so is the estimated uncertainty. This is a surprising result. We would expect that the increased local accuracy of the planar model leads to improved stability against linearisation errors.

The higher accuracy achieved by the planar feature setup can be attributed to two factors. First, subpixel accuracy is inherently provided by the direct measurement model. Second, including the normal vector into the state representation leads to improved appearance prediction. This can be verified by adding these capabilities to the point feature matching. The point measurements used above employ the pixel-accurate matching method. Now, we add subpixel refinement as described in Section 4.7.1. Finally, we

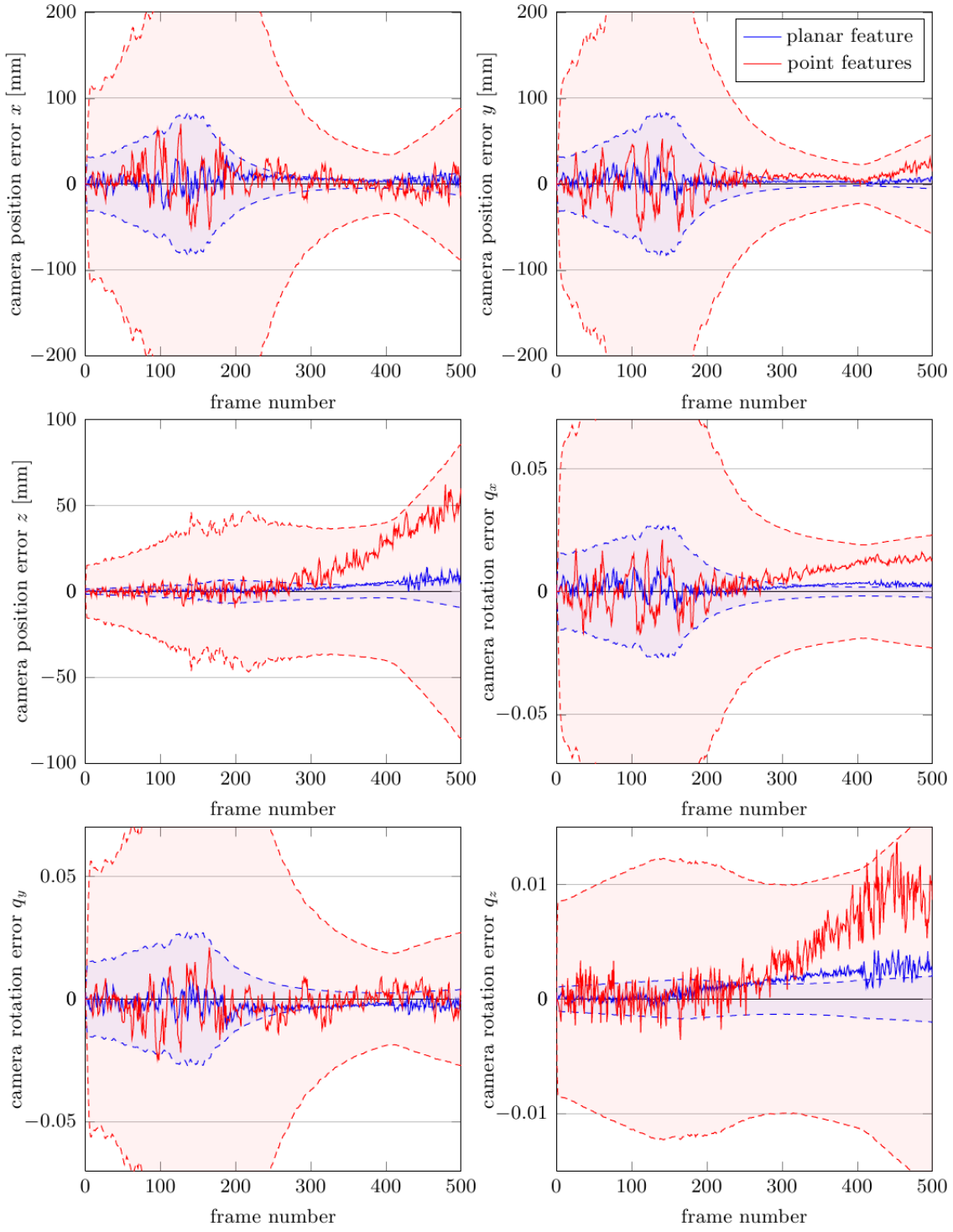


Figure 7.16.: Camera component errors for the "fur" sequence with 33 ms motion blur and $\sigma = 10$ intensity noise. The plot compares a single planar feature and three point features. 3σ uncertainty bounds are shown as dashed curves.

provide *a priori* correct normal vectors for the point features to judge the influence of accurate appearance prediction. We repeat the experiments with subpixel refinement, and with both, subpixel refinement and known normals.

Figure 7.17 illustrates the accuracy that is achieved by the planar direct intensity measurements in comparison with the above point feature measurement methods. The plots in the left column show the absolute measurement error over time for the planar feature and each of the three point features. The scatter plots in the right column illustrate the 2D distribution of measurement error in the reference image. The errors for the planar feature are shown in blue. Errors for the three point features are shown in red, orange, and green, respectively.

The top row of Figure 7.17 shows the results of the pixel-accurate matching method. Point measurement errors are in the order of 1 pixel, clearly much larger than for the planar measurement. The second row shows the results for the subpixel-accurate matching method. Point measurement errors are much smaller now. In the beginning they are comparable in accuracy to the planar measurement. Towards the end, systematic bias in the predicted appearance of the matching template causes systematic errors in the point measurements. Note, that in the scatter plot each of the point features shows an individual bias pattern according to the individual errors in the normal estimate. Finally, in the third row, point features are provided with ground truth, correct normal vectors. This removes the systematic bias. Accuracy is similar to the planar feature measurement throughout the sequence. The planar measurement is still slightly more accurate, which can be simply explained by the larger template.

Similar observations can be made for the camera pose estimate. Figure 7.18 compares the z camera translation estimate of the above point matching methods with that of the planar direct intensity measurement. For subpixel-accurate point measurements the estimated translation is smoother than for pixel-accurate measurements. Known normal vectors remove the remaining systematic bias.

The red curve in the plot at the bottom of Figure 7.18 shows the best possible accuracy that could be achieved with point features. Planar features and direct measurements provide accuracy that is very close to this baseline. Please note that for the planar feature, the normal vector is *estimated from the images* while it is *a priori* provided as ground truth for the point features.

Moreover, note that the point feature locations provide a slightly better triangulation baseline, which also helps to explain the remaining accuracy difference. Increasing the size of the planar feature provides more pixels for each measurement. This further increases the accuracy. We repeat the experiment with a larger planar feature of 131×131 pixels, which is indicated by the dashed green outline in Figure 7.13. The result is illustrated in Figure 7.19. The accuracy is improved over both the smaller planar feature and the point features with normals. Also note that the larger template improves the consistency of the estimate.

Noise and Motion Blur. For different settings of additive noise and motion blur, the results are similar to those presented above. The absolute camera error for noise settings of $\sigma \in \{0, 10, 20, 30\}$ is illustrated in Figure 7.20. The sequence contains no motion blur.

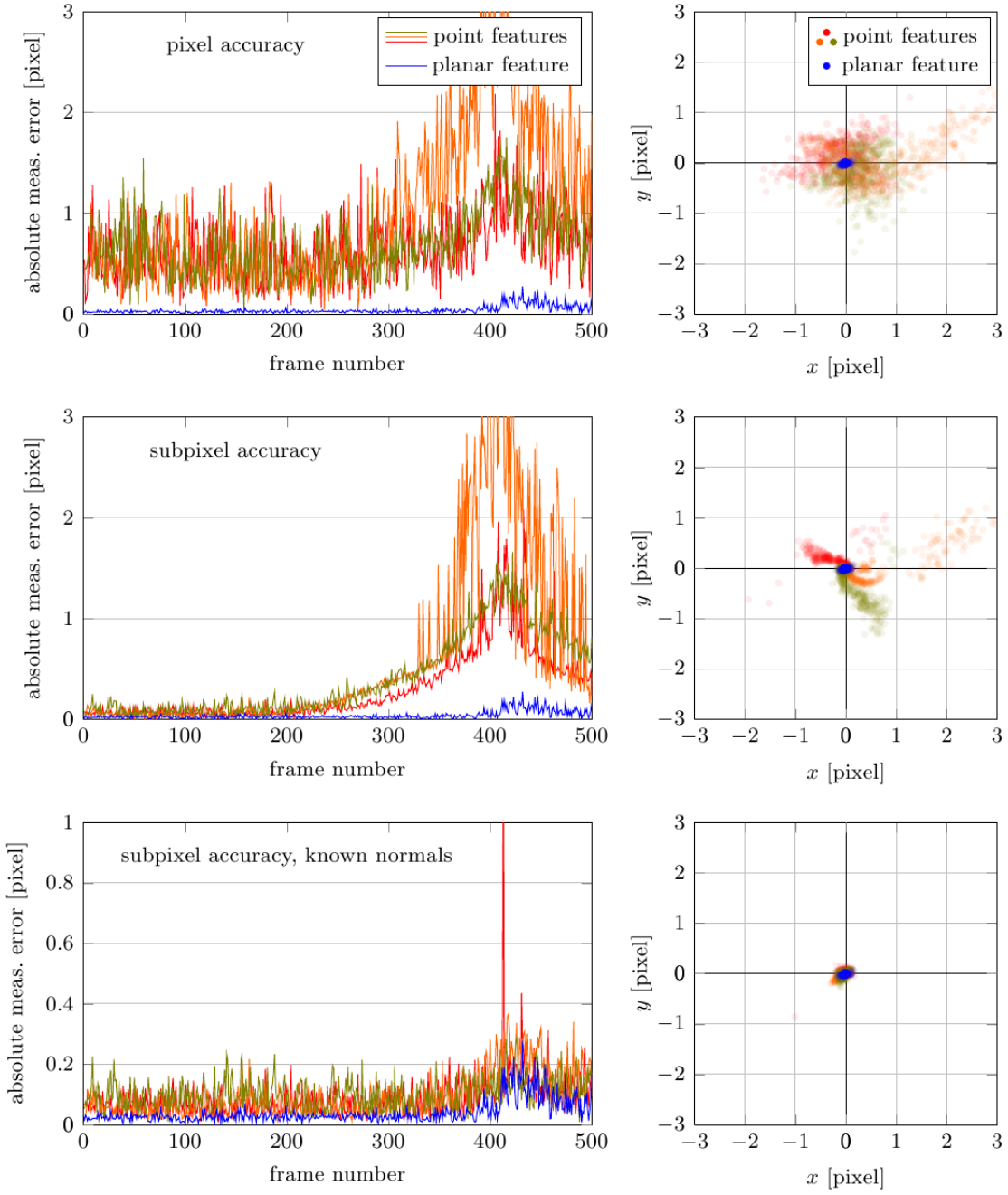


Figure 7.17.: Absolute errors and scatter plots for measurements in the "fur" sequence with 33 ms motion blur and $\sigma = 10$ intensity noise. In all plots, the blue curve shows the result of the planar feature. This is compared with point features using pixel-accurate measurements (top row) sub-pixel accurate measurements (middle row) and *a priori* known normals (bottom row).

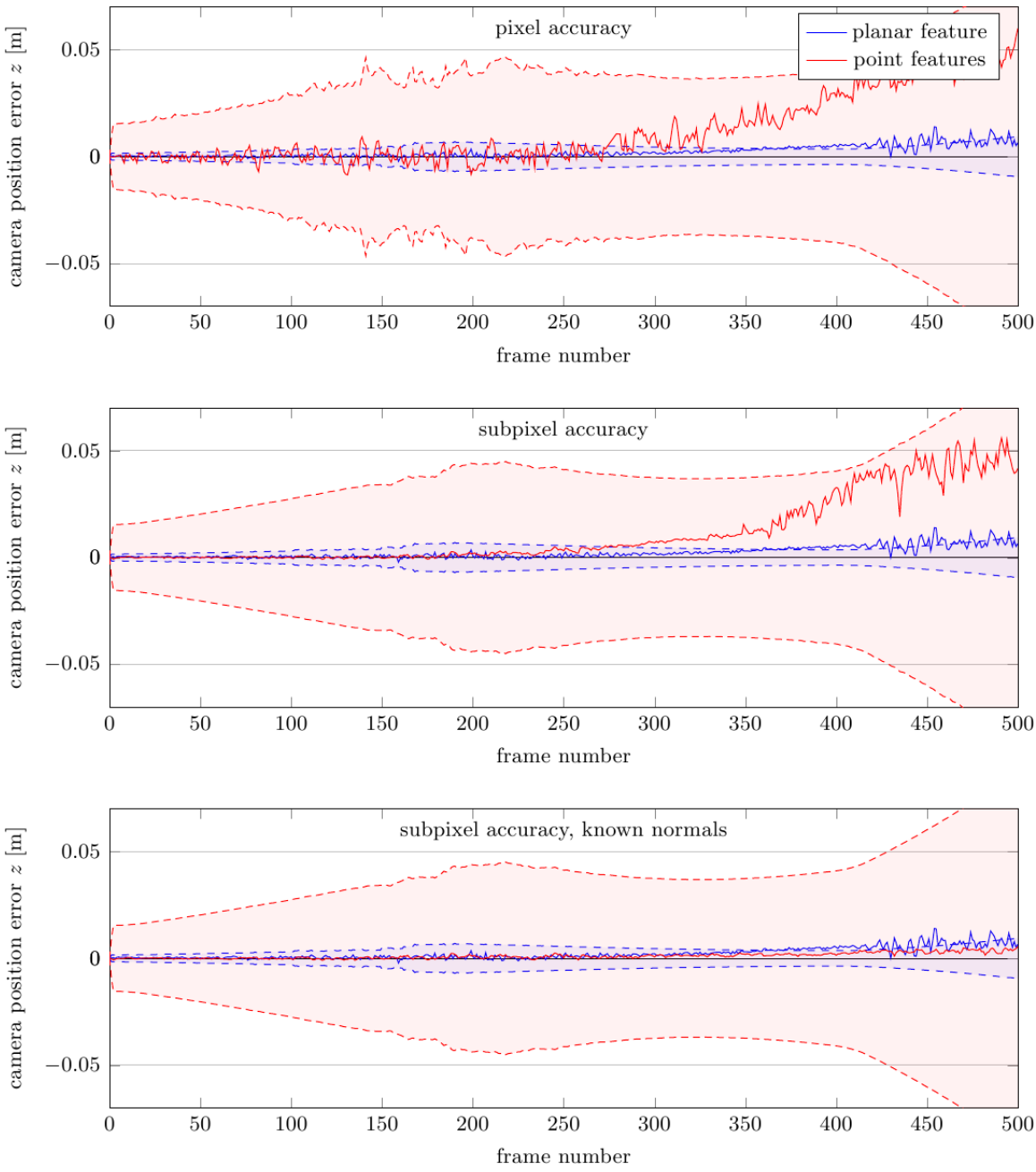


Figure 7.18.: Error in the z coordinate of camera translation for the "fur" sequence with 33 ms motion blur and $\sigma = 10$ intensity noise. In all plots, the blue curve shows the result of the planar feature. This is compared with point features using pixel-accurate measurements (top) sub-pixel accurate measurements (middle) and *a priori* known normals (bottom).

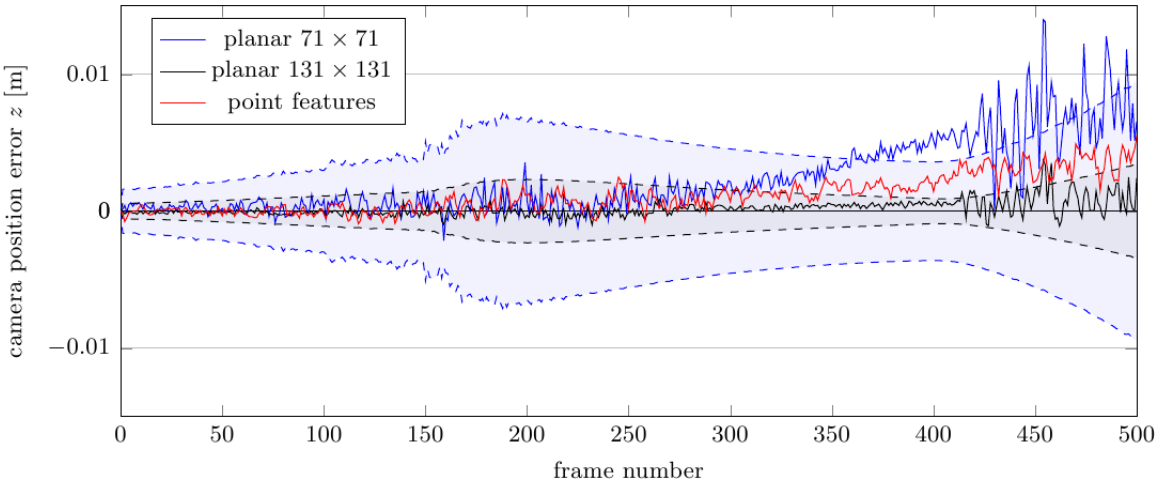


Figure 7.19.: Error in the z coordinate of camera translation for the "fur" sequence with 33 ms motion blur and $\sigma = 10$ intensity noise. A larger planar feature improves accuracy of the estimate. The point feature plot is for subpixel-accurate measurements with *a priori* known normals.

For the point features, subpixel-accurate measurement is used. The point features and planar feature setups both show degrading accuracy with increase noise level. For $\sigma = 30$, with point features, tracking is lost around frame 400. Similar tracking failures can be observed consistently for varying settings of motion blur and also for different choices of point feature positions. A likely explanation is that towards the end of the sequence the point feature templates are strongly foreshortened because of the shallow viewing angle. Consequently, the matching template is small and the ZSSD values are dominated by the pixel noise. The planar feature is more stable here simply because it has a larger template. Using larger templates for point features would improve tolerance to additive noise; however, it would also increase computation time for matching and increase the systematic bias caused by the incorrect normal estimate.

The absolute camera error for motion blur with shutter time 0 ms, 33 ms, and 99 ms is illustrated in Figure 7.21. The sequence contains no additive noise. The estimation error increases with increasing motion blur for both setups. For planar features, the effect seems to be more severe.

Scale Adjustment. We implemented the scale adjustment through anisotropic smoothing described in Section 7.8.3. This preprocessing step is intended to counter errors caused by image sampling artifacts. We compare estimated pose errors for the planar feature setup with and without this preprocessing step. The experiment was run on the "fur" sequence for all combinations of motion blur and additive noise, as described above.

In general, the preprocessing results are very similar to the raw planar feature. In almost all runs, preprocessing results in slightly smoother trajectory estimates and the absolute error is slightly decreased. The absolute camera position error for the experiment with $\sigma = 10$ additive noise and 33 ms motion blur is shown in Figure 7.22. It can be seen that

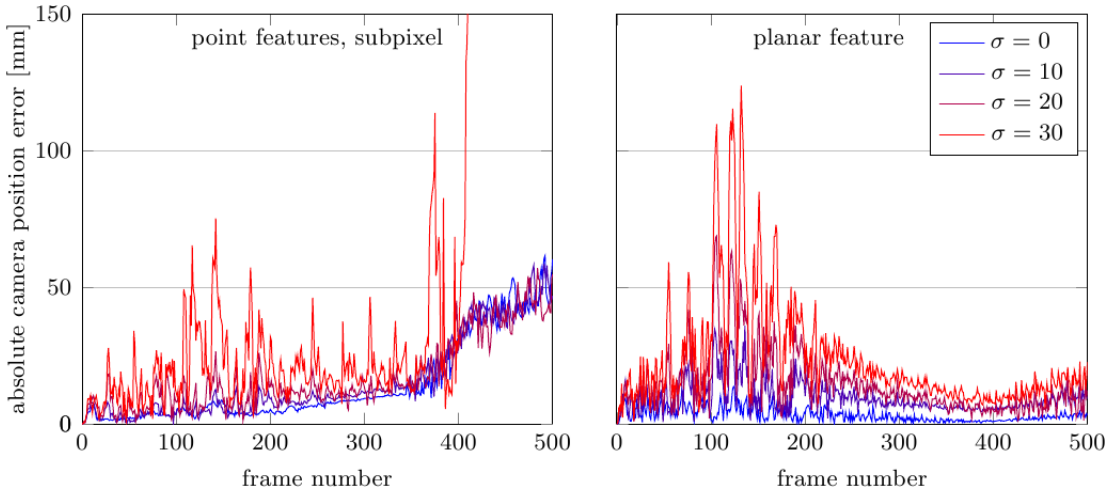


Figure 7.20.: Absolute camera position error for the "fur" sequence for varying levels of intensity noise, without motion blur. Results for point features with subpixel-accurate measurement are shown on the left. Results for the planar feature model are shown on the right.

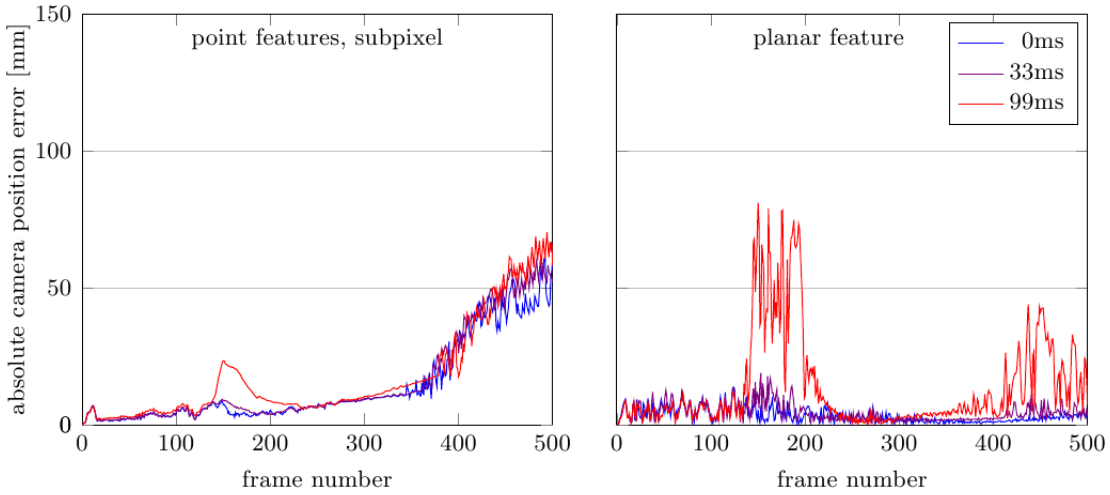


Figure 7.21.: Absolute camera position error for the "fur" sequence for varying settings of motion blur, without intensity noise. Results for point features with subpixel-accurate measurement are shown on the left. Results for the planar feature model are shown on the right.

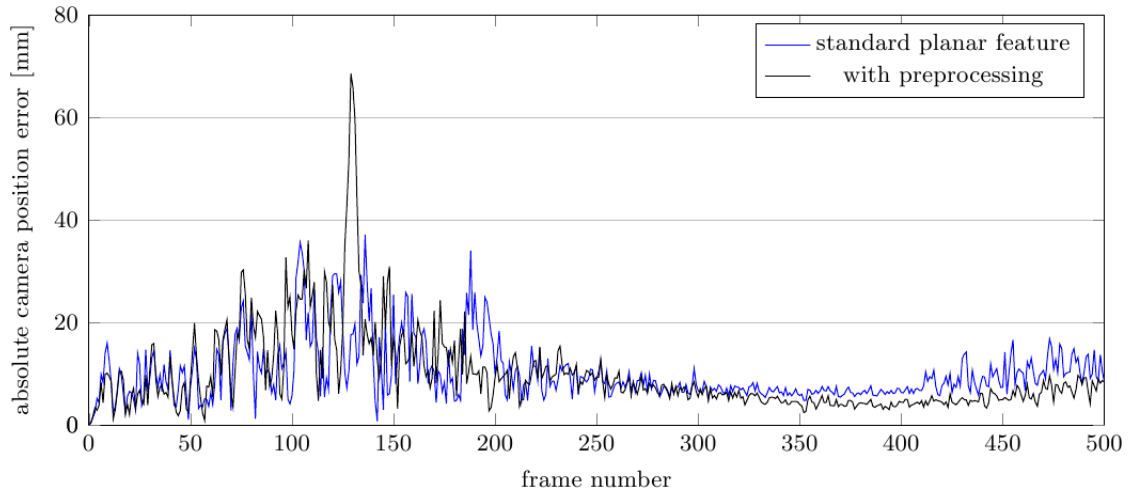


Figure 7.22.: Absolute camera position error for the "fur" sequence with 33 ms motion blur and $\sigma = 10$ intensity noise. The plot compares the standard planar feature model with and without an anisotropic smoothing preprocessing step.

the preprocessing step leads to slight improvements. The improvement is most significant towards the end of the sequence. There, the current camera viewing angle and the template camera pose are very different, causing severe distortion of the feature template. The improvement comes at the price of increased computational cost, though, because the preprocessing step involves smoothing the images with large anisotropic Gaussian kernels.

Multiple Features and Loop Closure

The next experiment evaluates the planar feature model on a more complex sequence. We build a full map of planar features. New features are initialised on-the-fly during the course of the sequence. Old features move out of the image and are re-acquired after closing a full 360° loop.

"Box" Sequence. This sequence already has been used in Chapter 6. See pages 135–136 for a full description and illustrations. The scene is a textured 6 m × 6 m × 6 m cube. The camera trajectory describes two full loops inside the cube. This is a long sequence comprising a larger environment and a more realistic camera motion. No part of the scene remains visible throughout the sequence. The SLAM system must close several loops. A reconstruction of the trajectory and map using planar features is shown in Figure 7.23.

Visual SLAM systems using planar or point features, respectively, are run on the sequence. For the planar setup, the feature outlines are selected manually as described above. The measurement process includes the scale adjustment preprocessing step. For the point feature setup, we use inverse depth bundle features in the visual SLAM framework described in Chapter 4. New features are selected and initialised automatically. The subpixel-accurate measurement method is used. In both setups, all visible features are measured in every frame.

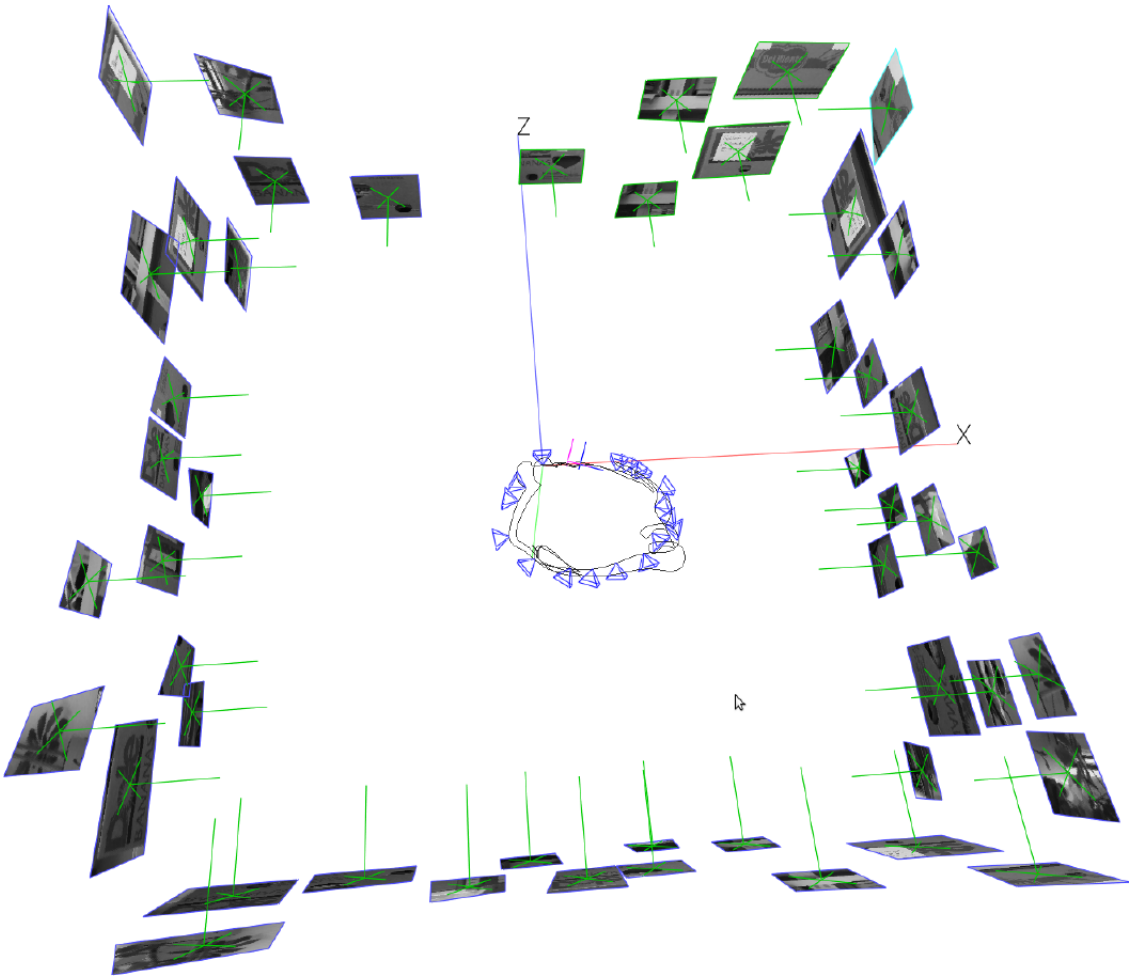


Figure 7.23.: 3D view of the reconstructed camera trajectory and map of planar features for the "box" sequence. The planar map provides a good sense of the structure of the scene.

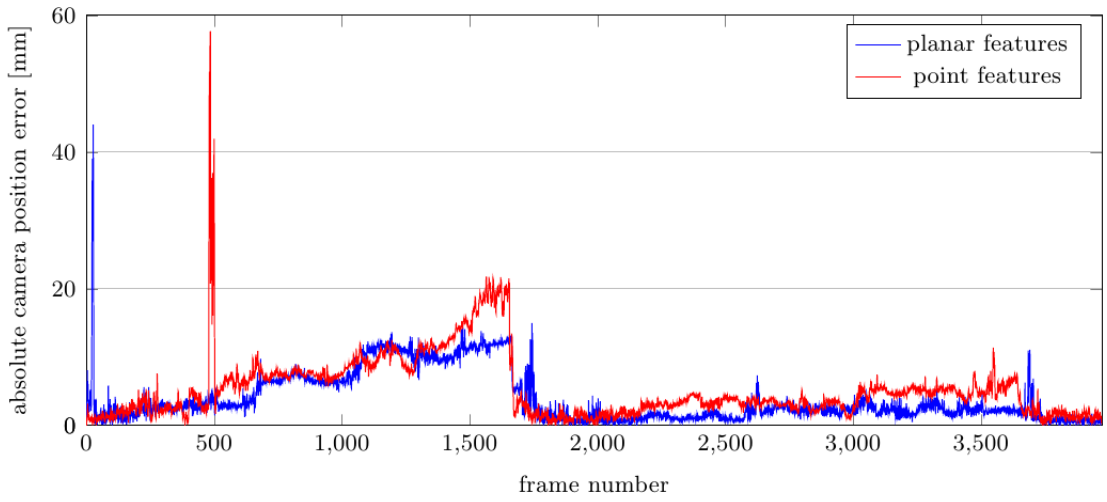


Figure 7.24.: Absolute camera position error for the "box" sequence with $\sigma = 10$ intensity noise, without motion blur.

As before, the experiment is performed for varying settings of motion blur and additive Gaussian noise. We first present results for the experiment without motion blur and with additive noise with standard deviation of $\sigma = 10$ intensity values. The absolute camera position error is shown in Figure 7.24. Errors are similar for both planar and point features. The first 360° loop closure around frame 1600 is clearly visible in the plot: In the exploratory phase before the loop closure, drift accumulates in the camera position error. After features are re-acquired, the estimate is corrected and position errors remain small as the loop is traversed for the second time. This can be observed for both, the planar features and point features setup. The overall accuracy of the reconstructed trajectory is slightly better for the planar features.

The best-fit map error is illustrated in Figure 7.25. Here, the planar model clearly outperforms the point feature model. At the end of the sequence, the error is 15.3 mm for point feature map while it is 2.5 mm for the planar feature map.

In general, we find, that image degradation by motion blur and/or intensity noise has a more severe effect on the planar model, confirming the observations of the "fur" experiment. We present results for the experiment with 33 ms motion blur and $\sigma = 20$ additive noise. The absolute camera position error is shown in Figure 7.26. Both, the point feature and planar trajectories are noisier. The overall magnitude of noise is also increased for both setups. In particular, note the large increase of error before the loop closure.

The best-fit map error is shown in Figure 7.27. While the map error is only slightly increased for the point feature model, the accuracy of the planar map has degraded considerably. In this experiment, the map error is about the same for both models.

Partially, the degraded map quality can be explained by the planar features estimate becoming overconfident. During the first loop up to frame 1600, a gradual build-up of planar map error can be seen. This is caused by accumulated camera drift during exploration of unmapped parts of the environment. Upon loop closure, the map can not be adequately corrected, because the EKF has become too confident in the estimated fea-

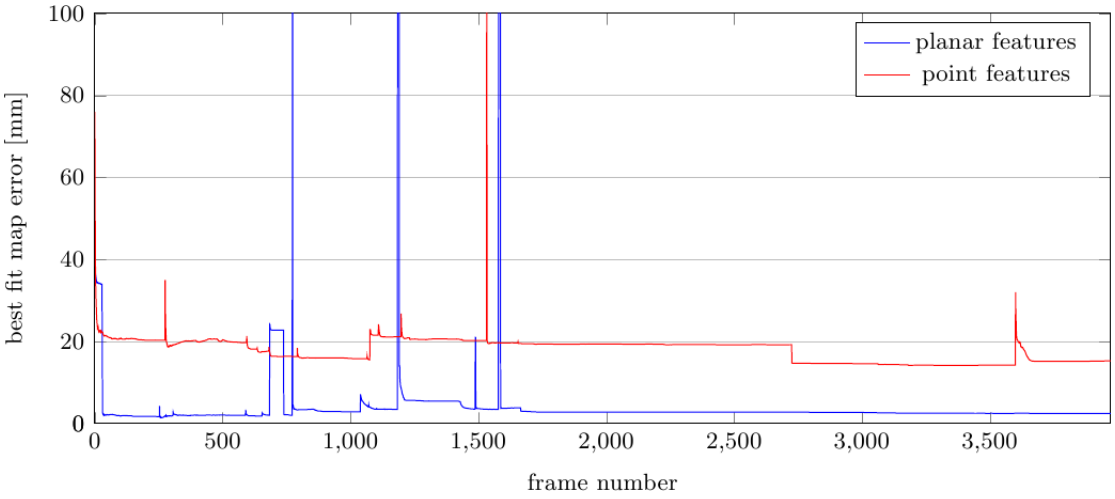


Figure 7.25.: Best-fit map error for the "box" sequence with $\sigma = 10$ intensity noise, without motion blur.

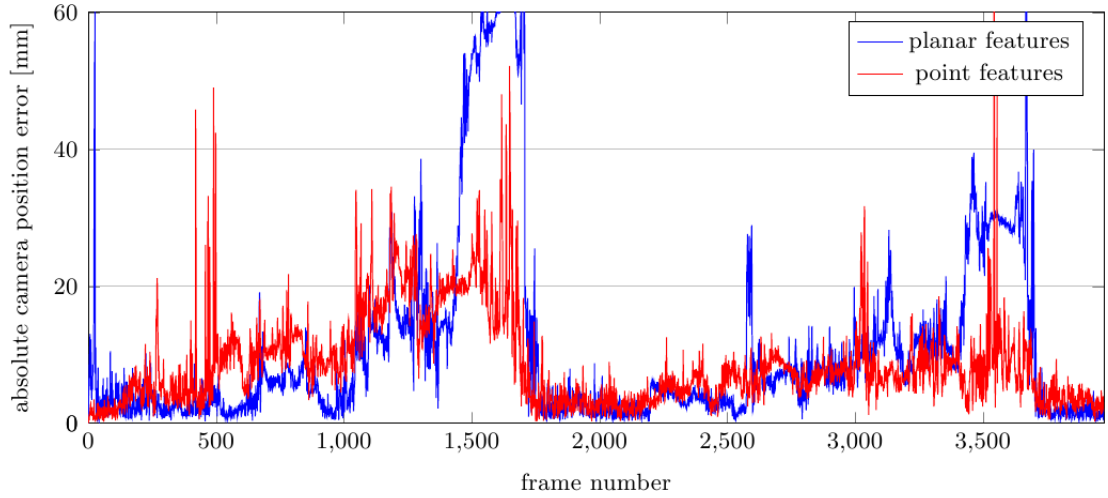


Figure 7.26.: Absolute camera position error for the "box" sequence with $\sigma = 20$ intensity noise and 33 ms motion blur.

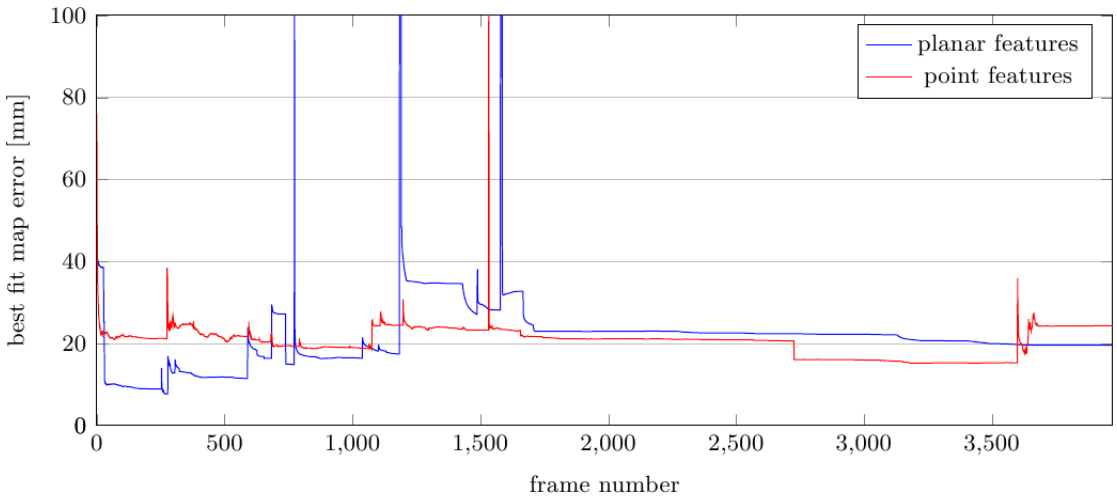


Figure 7.27.: Best-fit map error for the "box" sequence with $\sigma = 20$ intensity noise and 33 ms motion blur.

ture parameters. An indication for this is that the camera error pattern of the first loop is repeated during the second loop. Note the repeating pattern in frames $0 - 2000$ and $2000 - 4000$ in Figure 7.26. To a smaller extend, the same behaviour is visible for point feature as well.

In Figure 7.28, the y component of the camera translation estimate is shown along with the 3σ uncertainty bounds. This further illustrates the under-estimation of uncertainty. This is caused by both motion blur and additive noise artefacts. Motion blur introduces intensity errors that are correlated between pixels. These are not modeled in the feature measurements, which provides an explanation for the degradation with motion blur. Additive Gaussian intensity noise *is* modeled, though. A probable explanation for the degradation in this case are linearisation errors.

In summary, this experiment has shown that complex sequences with many features can successfully be handled using the planar model. Features are re-acquired after occlusion and loops are successfully closed. For image sequences with little noise, the planar feature measurements clearly outperform point features in terms of mapping and localisation accuracy. For images with high noise levels and blurring artefacts, planar feature performance degrades, though.

7.9.3. Evaluation on Real Image Sequences

In the following, we present results on two real image sequences. These experiments demonstrate that the proposed planar feature and measurement models are useful in SLAM scenarios under real-world conditions. The first sequence illustrates successful loop closure in an office environment. With the second sequence we demonstrate the reconstruction of complex scene structure. Like before, planar feature outlines are selected manually and provided as input to the SLAM system.

The "lab" sequence is recorded in an office environment. Some textured planar objects

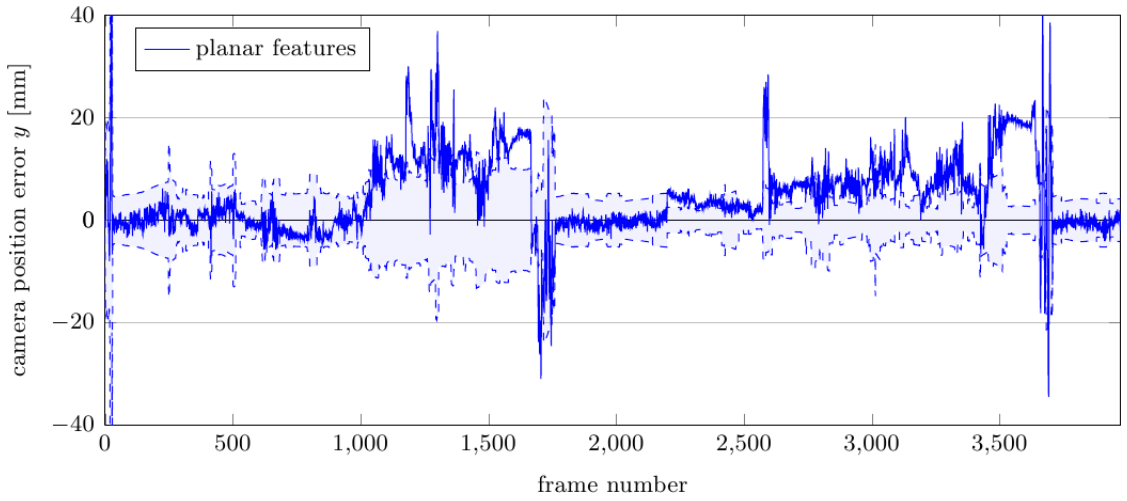


Figure 7.28.: Error in the y component of camera translation for the "box" sequence with $\sigma = 20$ intensity noise and 33 ms motion blur. The 3σ uncertainty bounds are indicated by the dashed curves.

have been added to the scene such that sufficiently many features are available for tracking throughout the sequence. The camera trajectory covers approximately 1 m × 1 m and comprises a 360° loop with the camera looking outward. Figure 7.29 shows selected images from the sequence. The projected outlines and normal vectors of planar features are overlaid on the images. The reconstructed map and trajectory are shown in Figure 7.30. The loop is successfully closed when features from the begin of the sequence are re-observed. We find the typical "snapping-into-place" of the map when accumulated drift is corrected upon loop closure.

Processing this sequence successfully required some effort to select reliable features by hand. Problems can occur in particular with features that are partially occluded. An example of partial occlusion is shown in Figure 7.31. In the course of the sequence, the feature outlined in red becomes occluded. On the left-hand side the figure contains one camera image where the feature is fully visible and one with the feature partially occluded. In this instance, the simple ZSSD-thresholding method described in Section 7.9.1 does not detect the occlusion. Instead, erroneous pixel measurements are incorporated in the state update. Here, the result is a shift in the estimated normal vector of the plane to compensate for the incorrect feature appearance. This is illustrated on the right-hand side of the figure.

To a surprisingly large degree, the SLAM system is able to track through such partial occlusions. However, if the occluded feature is the only one currently visible then tracking is likely to fail. The system can also tolerate some changes in image intensity such as caused (for instance) by small illumination changes. An example of a systematic intensity error is illustrated in Figure 7.32. Here, the cause of the variation is camera shading.[14]

[14]Shading, respectively vignetting, refers to the intensity fall-off towards the periphery camera image compared to the image center.

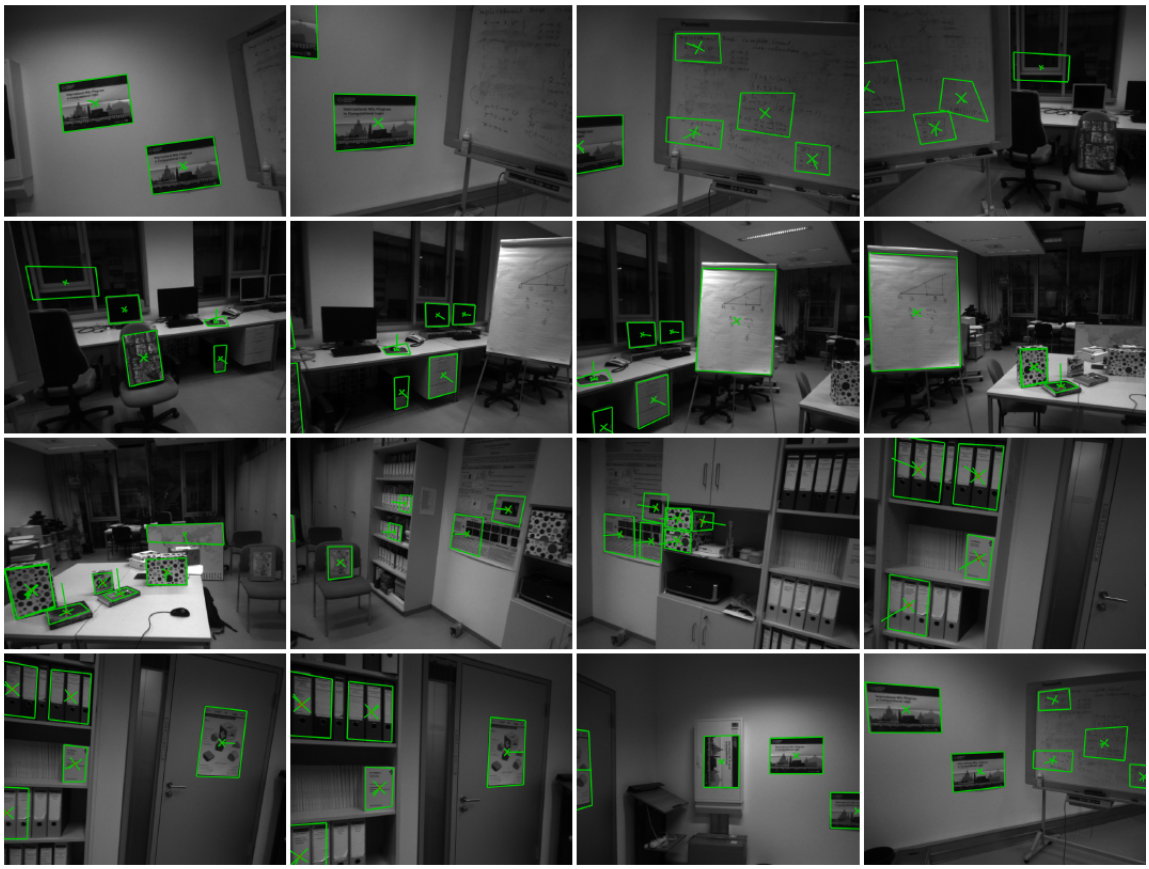


Figure 7.29.: Example images of the “lab” sequence. The projected outlines and normal vectors of mapped planar features are overlaid on the images.

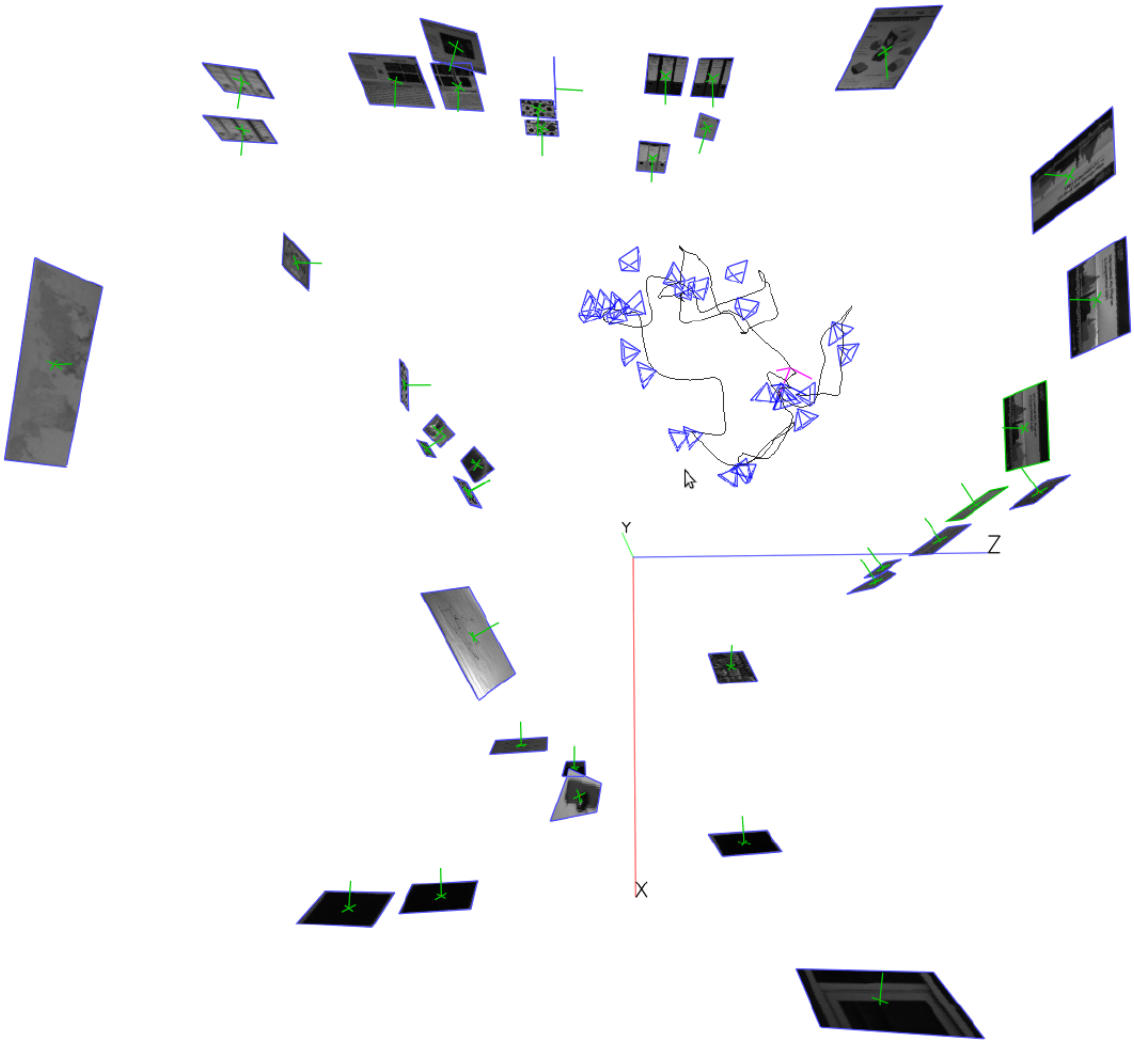


Figure 7.30.: Bird's eye view of the reconstructed camera trajectory and map of planar features for the "lab" sequence.

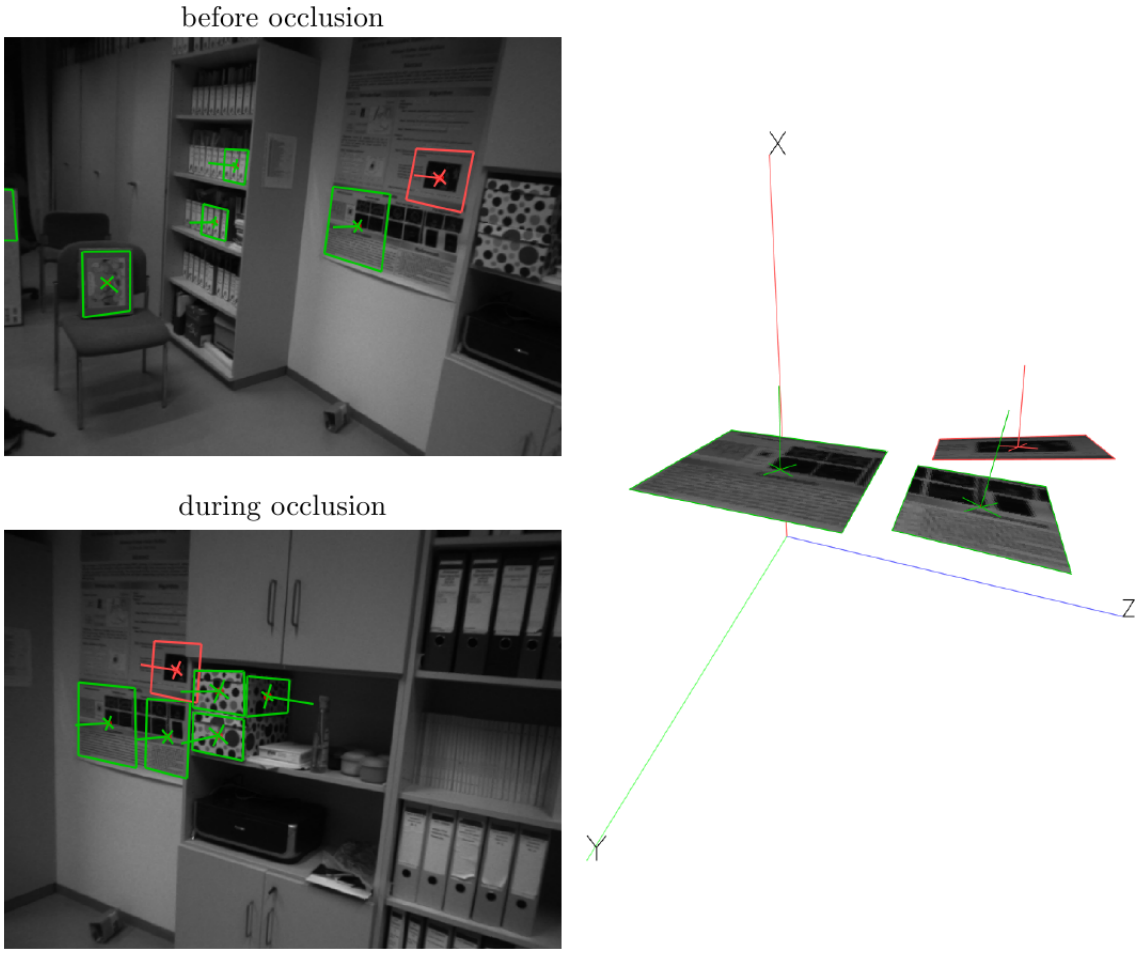
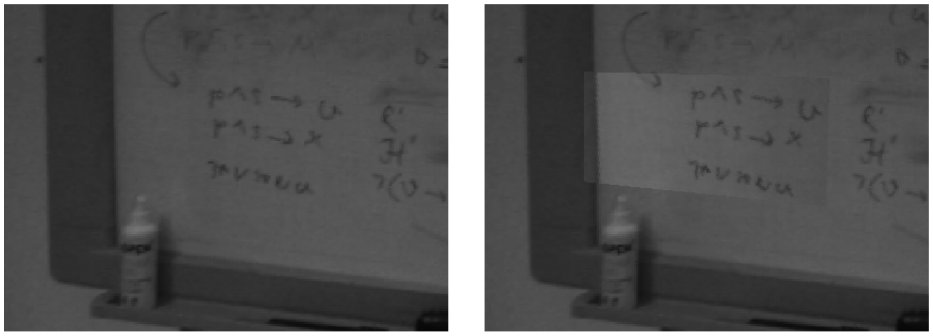


Figure 7.31.: Partial occlusion in "lab" sequence. Partial occlusion of the planar feature outlined in red cause erroneous pixel measurements. This results in an incorrect estimate of the normal vector.



(a) (b)

Figure 7.32.: Shading effects cause systematic intensity biases. (a) shows a detail of the current input image. In (b) the texture of the planar feature is overlaid on the image.

The texture of a planar feature is shown overlaid on the current input image. Clearly, there is a systematic bias in the respective intensities. Although such systematic errors are not currently modelled in our pixel measurement model, they are quite well tolerated. A likely explanation is that the highest information content is in pixels on intensity edges and there the systematic bias is small with respect to the edge intensity difference. In contrast, pixels in areas of largely uniform intensity exert little influence on the state estimate.

The current planar SLAM implementation is not able to process the "lab" sequence in real-time. Per-frame processing times are illustrated in Figure 7.33. The average update time per stereo image is 70 ms. A large part of this time, on average 37 ms, is spent on iterating the intensity measurement update. The second expensive operation is the initial ZSSD feature search, which requires 30 ms. The efficiency of both these steps could be increased. For the initial exhaustive search, it is wasteful to employ the full and potentially very large planar feature templates in ZSSD matching. Instead, a few interest points could be picked from the feature texture and searched in the image. For the intensity measurements, it would be useful to selectively measure only pixels with large intensity gradient. Pixels in uniform image areas provide no information and hence should be excluded from the measurement.

For the intensity update, on average 43 220 individual pixel intensities are measured in each image. The Jacobians of these individual pixels must be computed, aggregated into fused measurements, and employed in the IEKF iteration. IEKF convergence occured after 3.3 iterations on average. Thus, the current implementation can process more than 2 million pixel intensities per second. This number could be further increased by off-loading pixel Jacobians computation and aggregation to the GPU, as these operations can be naturally parallelized.

The final planar map for the "lab" sequence contains 38 planar features. In contrast, our bundle feature system describe in Chapter 6 creates a map of 181 point features for the same sequence. Thus, a reduction in map size is achieved, while at the same time creating a more descriptive and dense map.

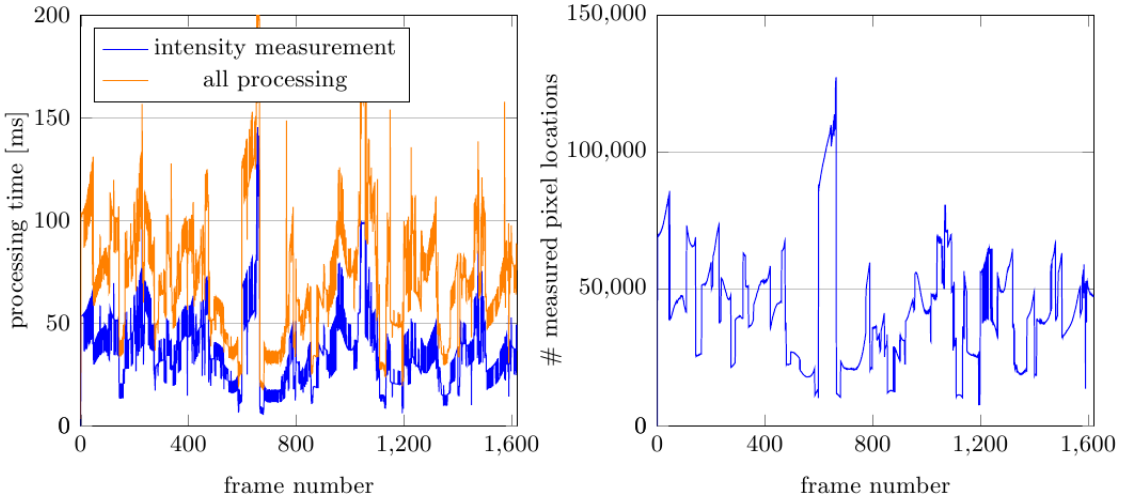


Figure 7.33.: Processing times and measurement size for the planar feature SLAM system on the "lab" sequence. The plot on the left illustrate the per-frame processing times. The orange curve shows the overall time required for every image, including image acquisition, prediction, update, and visualisation. The blue curve shows the time required for the iterative instensity measurement step, which takes up a large part of the overall processing time. The plot on the right shows the number of pixels of the current stereo image that participate in the intensity measurement.

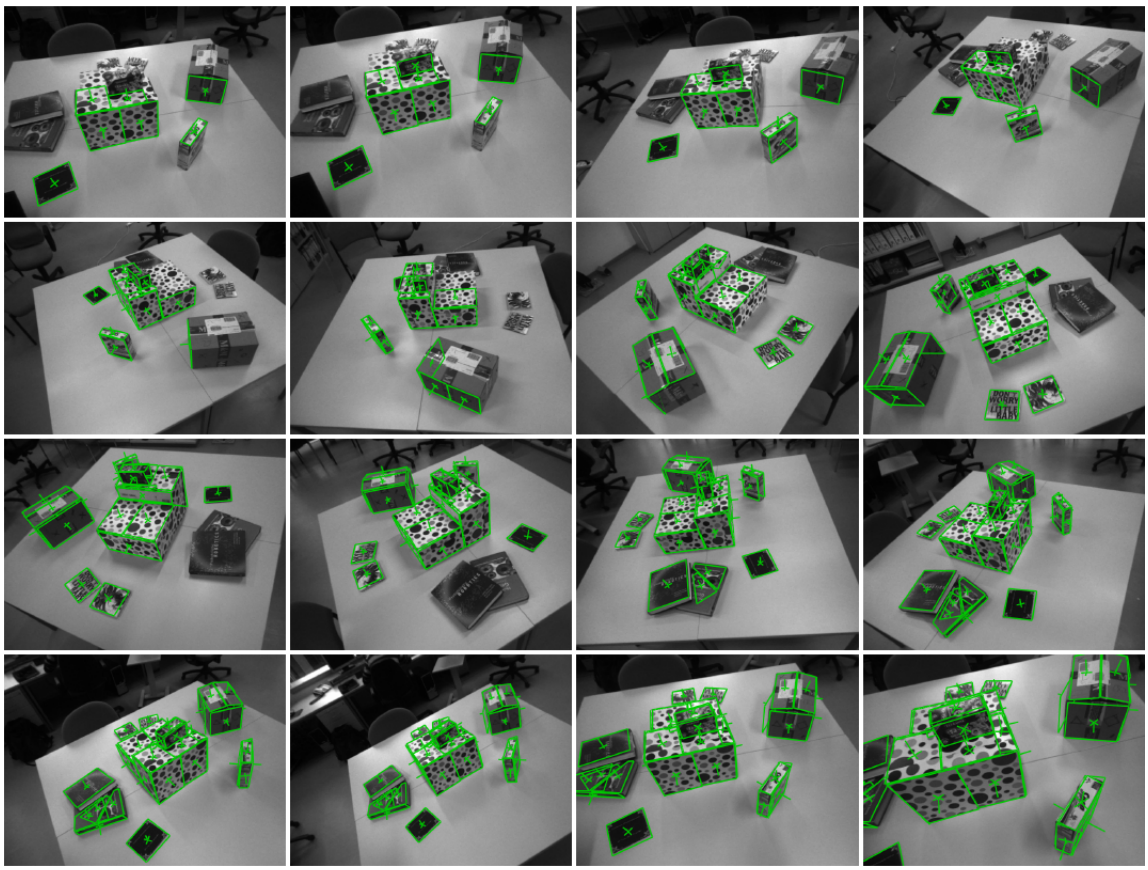


Figure 7.34.: Example images of the "tabletop" sequence. The projected outlines and normal vectors of mapped planar features are overlaid on the images.

The next experiment uses the "tabletop" sequence. With this sequence, we want to demonstrate that planar features allow pseudo-dense reconstruction of non-trivial scenes. Several piecewise planar objects have been placed on a table. The camera is moved a 360° loop around the table, looking towards the table. In this way, all sides of the objects are visible at some point in the sequence.

Sample images from the sequence are shown in Figure 7.34. The projected outlines and normal vectors of planar features are overlaid on the images. As before, feature outlines are initialised manually. Note that various partial, full, and self-occlusions occur. Nevertheless, the sequence is successfully tracked. Occlusions are often detected by simple ZSSD thresholding method described above. Figure 7.35 shows views of the reconstructed map. The map contains minor artefacts caused by partial occlusions. It gives a good representation of the scene, though. For instance, the scene structure and appearance is immediately accessible to human interpretation.

The pseudo-dense planar map can also be useful for computing occlusions in augmented reality applications. In Figure 7.36 virtual teapots are inserted into the images of the "tabletop" sequence. The known geometry of the mapped real surfaces allows to render

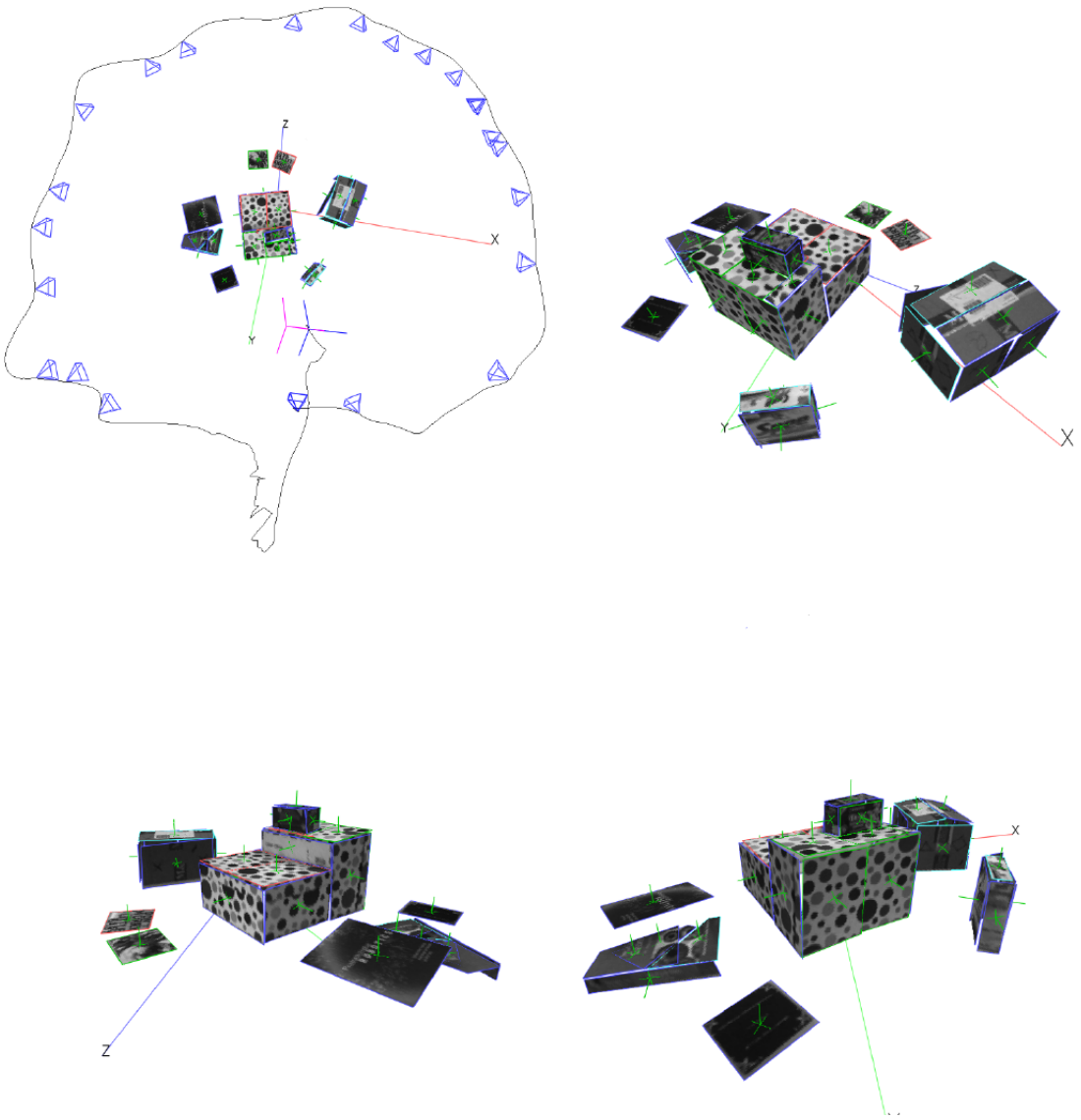


Figure 7.35.: Views of the reconstructed map of planar features for the "tabletop" sequence.

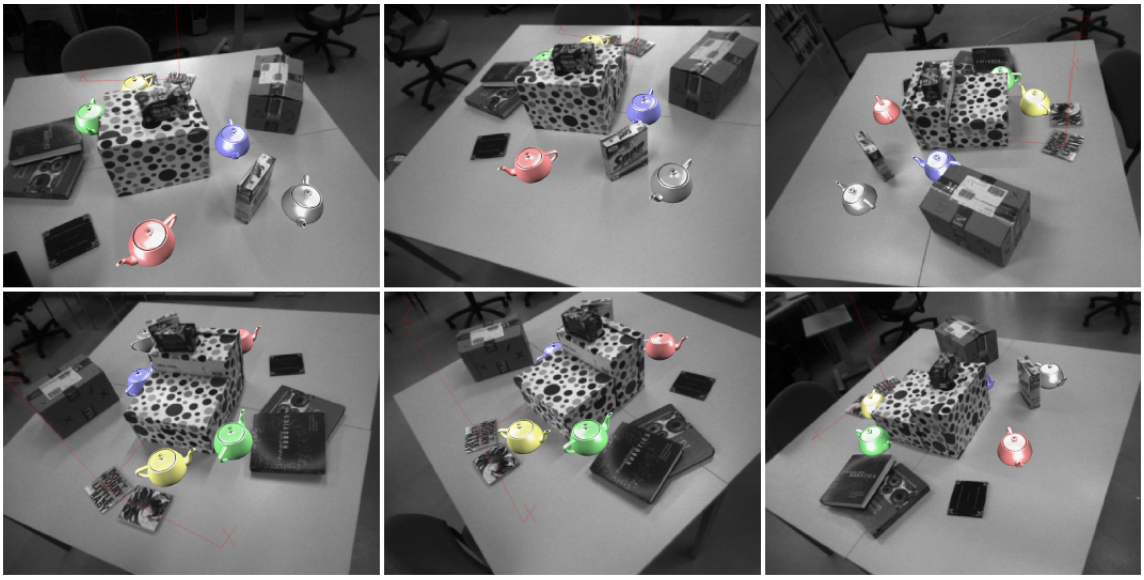


Figure 7.36.: Real-virtual occlusion using the reconstructed planar map of the "tabletop" sequence. Augmented virtual objects are rendered such that they appear correctly occluded by real objects

only visible parts of the overlaid objects such that they appear to be occluded by real objects in front of them.

7.10. Discussion

We have presented an approach of incorporating planar structures as features in EKF based visual SLAM. Planar features represent planar surface segments in the environment. The plane on which a feature lies is represented with respect to a reference camera frame. The feature extent and appearance are defined by back-projecting the feature outline and texture, respectively, from the reference image onto this plane. Analogous to view-point based point features, the explicit representation of the reference camera pose (anchor) is crucial for the accurate prediction of feature appearance.

We have proposed a direct measurement method for planar features. In our measurement model, measurements are high-dimensional vectors of pixel intensities in the current image. Thus, information provided by changes in feature appearance due to changing camera view-point is directly utilised to improve the state estimate. This measurement approach is reminiscent of methods used in image alignment. Here, for the first time, such an approach was integrated into full visual SLAM. The result is a near-real time EKF based SLAM system using a fully correlated map of planar features.

Applying direct measurements in this scenario required several new contributions. We proposed an iterative measurement update step to deal with the high nonlinearity in the functional relationship between the image intensities and camera and feature pose. The iterative update is initialised using a reduced point-feature-like measurement model and

exhaustive template search. The initial solution is iteratively refined using an IEKF. The iterations of the IEKF are interleaved with predicting feature outlines and measuring pixel intensities in the current image. During the iterative refinement, not only the camera and feature positions are updated but also the normal vectors of the features.

We further proposed a general, principled method to handle high-dimensional measurements in the EKF framework. A planar feature measurement usually comprises thousands of individual pixel intensities. With the cubic dependence of the EKF update step on the measurement size, this presents a problem, especially in a system targeting real-time performance. We employed *measurement fusion* to reduce the complexity of the update step from cubic to linear. The effectiveness of the method was demonstrated in our experiments: Measurement vectors on the order of 50 000 pixels are handled in near real-time.

The direct intensity measurement allows to extract more information from the images than is available from traditional 2D point feature measurements. Specifically, it allows to measure the homography induced by the feature plane between the reference and current camera images. This, in turn, fully constrains the relative pose of feature and camera. We have shown that a single planar feature is sufficient for tracking the camera pose.

The planar feature model has been evaluated using both rendered and real image sequences. In simulation experiments using the SLAMDUNK framework, we evaluated the accuracy of planar features in comparison to point features. We found that planar features in conjunction with direct intensity measurements allow high accuracy estimation of camera trajectory and map. The reconstruction accuracy rivals that which can be obtained by subpixel-accurate point feature matching. This is even the case if point feature templates can be accurately predicted using *a priori* known normal vectors. For this comparison we have examined image sequences with different amounts of motion blur and intensity noise. For images with little distortion, the planar features surpass the accuracy of point features. As blur and noise distortions increase, this advantage is lost and the planar features perform similar to point features. We further demonstrated planar feature SLAM on real image sequences including loop closures, full and partial occlusions, and systematic intensity errors.

The ultimate goal we pursue with this work is to create dense environment maps. The planar feature model presents a first step in this direction. Planar features allow pseudo-dense maps, where a dense representation is provided for areas of the scene that comprise textured planar surfaces. Moreover, these parts of the scene can be represented more efficiently than with point features. This is illustrated for instance by the "tabletop" experiment. Here, the map comprises only 39 planar features. However, this gives a high-quality map that is easily human-comprehensible and useful for advanced applications, such as augmented reality occlusion rendering.

Clearly, planar features alone will not be sufficient to build fully dense maps as there are always parts of the environment that are non-planar or texture-less. In this sense the planar model presents exploratory work that shows how dense information can be incorporated into visual SLAM systems. Importantly, we have shown that this can be achieved in a fully integrated approach. Instead of layering dense reconstruction on top of a point-based SLAM system, in our case the dense representation *is* the SLAM map. Softening the planarity requirement and extending the approach to more general surfaces is a challenging topic for future research.

Several open issues remain in the presented planar SLAM system. First, a method for automatic detection of new planar features is required. For this purpose, planar and point features should be combined. In an approach similar to (Martínez-Carranza & Calway, 2009b), a combined system could immediately start tracking with point features and use the camera pose estimate to sequentially detect planar structure.

A further remaining problem are very distant features. For these features, the stereo baseline, as well as the camera motion during the initialisation phase is very small in comparison to the feature depth. This means that there is little observable parallax within the feature area, and consequently almost no information is observed about the feature plane orientation. In such cases, the feature normal estimate artificially converges to incorrect and often extreme values. Similar problems have been observed by Eade & Drummond (2006b) with respect to artificial landmark depth convergence. Their solutions is to discard measurements of new features if they do not provide sufficient parallax information. A similar strategy could be beneficial here.

Another challenge is to increase the robustness of the intensity measurement process, in particular the handling of partial occlusions. Developing methods similar to JCBB, RANSAC, or Active Matching, that enforce global consistency on the level of pixel measurements is a very challenging and interesting open problem.

Moreover, with respect to computational cost, superfluous work is expended by integrating measurements of pixels in areas of (almost) uniform intensity. These pixels contribute only little to the measurement. Truly informative pixels are those on strong intensity edges. Thus, the computation speed of planar SLAM would benefit from selective pixel integration (Buenaposada & Baumela, 2002b; Dellaert & Collins, 1999).

8 Conclusion

8.1. Summary

In this thesis, we have been concerned with the mapping aspect of visual SLAM. We have proposed feature representations that allow a more dense description of the environment. Specifically, we have discussed the inverse depth bundle representation for point features and the planar feature representation for textured plane segments. To handle image measurements of plane features a novel iterative update procedure and a method to efficiently fuse high-dimensional measurement vectors have been introduced. To empirically validate the proposed models thoroughly we have developed an automated evaluation framework for visual SLAM.

The framework presented in Chapter 5 is used to evaluate the performance of visual SLAM systems on rendered image sequences. Similar to the commonly used point cloud simulations, reliable and accurate ground truth is available in rendered image sequences. However, rendered images allow to examine details below the level of abstraction of point clouds. In this thesis, this has been invaluable in the evaluation of measurement bias effects. The method allows evaluation of SLAM systems that do not employ point features, such as the proposed planar feature system. Moreover, the evaluation framework does not interfere in any way with the operation of the visual SLAM system under evaluation. In particular, the SLAM system automatically selects visual features in the images. The framework creates ground truth data accordingly. Rendered images have been successful tools in standardised benchmarks in computer vision, e.g., in the evaluation of stereo reconstruction algorithms. The proposed evaluation framework may help to devise similar benchmarks for the field of visual SLAM.

The inverse depth bundle parameterisation has been proposed in Chapter 6. This representation significantly increases the number of point features that can be handled in real-time EKF based systems. The bundle parameterisation achieves this by grouping features by their common frame of initial observation. This allows parameter sharing be-

tween features which reduces the size of individual features in the state vector. We have shown experimentally that fully correlated maps of more than 200 features are handled in real-time. This is four times the size of what can be achieved using the popular unified inverse depth parameterisation. The performance gain is possible because the bundle representation moves two degrees of freedom of every feature into the common anchor representation. We have experimentally shown that the effects on accuracy are negligible.

The planar feature model as well as a direct measurement method for such features have been presented in Chapter 7. Planar features represent textured planar surface segments in the scene. The texture and the outline of a features are defined with respect to a reference image. All parameters of the plane are estimated from image measurements, including the normal vector. A planar feature can compactly represent large and arbitrarily shaped planar segments. This allows to reconstruct maps that efficiently represent dense structure for piecewise planar scenes. In contrast to approaches that layer dense reconstruction methods on top of point-based SLAM systems, we rely on a fully integrative approach. The dense reconstruction *is* the map. It is used for localisation and refined over time. Full correlations are maintained between planar features allowing the dense reconstruction to benefit from loop closure constraints, for example.

We have used a direct measurement method for planar features, where pixel intensities are employed directly as measurements. To deal with the high non-linearity of this measurement model we have proposed a novel measurement procedure using iterative refinement. The pixel measurement process is interleaved with the update loop of an Iterated Extended Kalman Filter. This iterative process is initialised using a simplified point-feature-like measurement model, thus providing increased robustness to large image motion.

We have further proposed a principled approach to reduce high-dimensional measurements to equivalent fused measurements of bounded dimension. Given reasonable and common assumptions about the measurement model, the method reduces the EKF update complexity from cubic to linear in the measurement dimension. This measurement fusion approach has been crucial to achieving near real-time performance in our planar feature implementation where each measurement comprises tens of thousands of pixels.

Both the inverse depth bundle and the planar feature model share a common underlying view of visual feature measurements. We emphasise the indirect nature of feature matching. Feature measurements are only ever achieved by establishing correspondence between images. The process can be viewed as the back-projection of a feature template onto the scene surface and subsequent projection into the current camera image. The feature template is established by the initial observation of the feature. Thus, it is appealing to view the initial observation not as a measurement in itself but merely as establishing the basis for future measurement. This insight has been crucial in devising the planar measurement model and in the analysis of measurement biases in the bundle parameterisation.

8.2. Future Work

Many possibilities to improve and extend upon the work in this thesis remain. Topics for immediate future work pertaining our contributions have been proposed already in

each of the Chapters 5 to 7: For automated evaluation, an important issue is to find a measure of overall map quality that includes accuracy and estimated uncertainty, as well as utility of the map. The inverse depth bundle approach could benefit from improved map management strategies that reduce the number of anchor frames. The work on planar features should be extended by automatic detection of new planar features. The robustness of the intensity measurement method should be improved by explicitly handling partial occlusions and illumination changes.

The map models proposed in this thesis have been implemented and tested using a stereo camera. It would be desirable to employ the models to the monocular case as well. In a monocular setting, the initialisation of new features will require further attention. With an inverse depth representation, point features may be simply initialised at a standard distance with a sufficiently large depth uncertainty. It would be interesting to see whether a similar approach is sufficient for planar feature initialisation.

Likewise, the application of the proposed models in estimation frameworks other than the EKF is an interesting issue to explore. Recently, visual SLAM systems based on keyframe bundle adjustment (e.g., Strasdat et al., 2010) have attracted much attention. Employing planar features and direct intensity measurements in this setting is certainly an intriguing possibility. Efficient feature parameterisation is less critical in bundle adjustment systems for the following reasons. In contrast to the EKF, the complexity of bundle adjustment is linear in the number of features. Moreover, by decoupling tracking and mapping, these systems do not require a map update at frame-rate. Nevertheless, it might be worthwhile to compare a point feature model with one degree of freedom – beeing similar to the inverse depth bundle model – to the three-parameter models commonly applied.

We have argued in this thesis that dense mapping is an important future direction of progress in visual SLAM. The presented planar feature model allows to build maps that comprise dense information for well-textured planar structures in the environment. Clearly, methods that allow dense mapping of more general environments are needed. Whether and how the presented approach can be extended to describe more general surfaces is an interesting question.

Finally, a promising direction of work is to exploit parallelism in the planar feature measurements. The described process of aggregating tens or hundreds of thousands of pixel intensities into fused measurements is very well-suited to parallelization. Given the current trend towards parallel hardware such as multi-core CPUs and many-core GPUs, we think that similar direct measurement methods will become increasingly attractive. These methods are very appealing from a technical point of view: They rigorously apply top-down modeling right down to the raw image. They provide the prospect of leveraging information from every single pixel in every camera image.

A
Jacobians

In this appendix we give the Jacobians (matrices of partial derivatives) of the process and measurement models presented in this thesis. These Jacobians can be worked out by hierarchic decomposition using the chain rule

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial x}. \tag{A.1}$$

The full Jacobians can then be written as matrix products of the Jacobians of a few common operations, e.g., pinhole projection or quaternion multiplication. Once these building blocks have been worked out, computing the model Jacobians is straightforward. To illustrate the idea, in Section A.1 we will in detail look at the Jacobians of the measurement model for Euclidean point features (as discussed in Section 4.6). In Section A.2, we will derive the Jacobians of some common functions (such as the projection discussed above). In the remainder of this chapter we will use these building blocks, to derive the Jacobians of the models presented in the thesis.

A.1. Introduction: Jacobians of Euclidean Measurement Model

Lets consider the measurement of a single point feature $\mathbf{y}_i$ that is parameterised by its Euclidean XYZ coordinates. The measurement function is, cf. Equation 4.17,

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{y}_i^{\mathcal{I}} + \boldsymbol{\delta} = \pi_s \left(p^{\mathcal{CW}}(\mathbf{y}\,; \mathbf{x}_v) \right) + \boldsymbol{\delta}. \tag{A.2}$$

We want to compute the derivative of this function by the state vector $\mathbf{x}$. Because the noise vector $\boldsymbol{\delta}$ is simply an additive constant, we have

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{x}}. \tag{A.3}$$

Let us first review how the image projection $\mathbf{y}_i^{\mathcal{I}}$ is obtained:

1. Transform the feature coordinates $\mathbf{y}_i$ from the world coordinate system to the current camera coordinate system, yielding $\mathbf{y}_i^{\mathcal{C}} = p^{\mathcal{CW}}(\mathbf{y}_i\,;\mathbf{x}_v)$. The transformation $p^{\mathcal{CW}}$ is parameterised on the current camera state $\mathbf{x}_v$. In fact, it is the inverse of the camera pose $p^{\mathcal{WC}}$ that is described by the state variables $\mathbf{q}^{\mathcal{WC}}$ and $\mathbf{r}^{\mathcal{WC}}$.
2. Project $\mathbf{y}_i^{\mathcal{C}}$ to the camera image plane, obtaining $\mathbf{y}_i^{\mathcal{I}} = \pi_s\left(\mathbf{y}_i^{\mathcal{C}}\right)$.

Writing this in full detail, we have

$$\begin{aligned}\mathbf{y}_i^{\mathcal{I}} &= \pi_s\left(\mathbf{y}_i^{\mathcal{C}}\right)\\ &= \pi_s\left(p^{\mathcal{CW}}(\mathbf{y}_i\,;\mathbf{x}_v)\right)\\ &= \pi_s\left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathbf{y}_i - \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathbf{r}^{\mathcal{WC}}\right)\\ &= \pi_s\left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathbf{y}_i - \mathbf{r}^{\mathcal{WC}}\right)\right).\end{aligned} \tag{A.4}$$

Now, let us note that only the $\mathbf{r}^{\mathcal{WC}}$, $\mathbf{q}^{\mathcal{WC}}$ and $\mathbf{y}_i$ components of the state vector $\mathbf{x}$ occur in the measurement function. Partial derivatives by the other state variables are $\mathbf{0}$ which gives the following sparse layout for the Jacobian

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{x}} = \left[\frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}} \quad \frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{q}^{\mathcal{WC}}} \quad \mathbf{0} \quad \cdots \quad \frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{y}_i} \quad \mathbf{0} \quad \cdots\right]. \tag{A.5}$$

Lets consider the first block, $\frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}}$. Applying the chain rule, we obtain

$$\frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}} = \frac{\partial \pi_s\left(\mathbf{y}_i^{\mathcal{C}}\right)}{\partial \mathbf{y}_i} = \frac{\partial \pi_s\left(\mathbf{y}_i^{\mathcal{C}}\right)}{\partial \mathbf{y}_i^{\mathcal{C}}} \cdot \frac{\partial \mathbf{y}_i^{\mathcal{C}}}{\partial \mathbf{y}_i}. \tag{A.6}$$

That is, the Jacobian decomposes into the product of two terms. The first term is the Jacobian of the projection function $\pi_s(\cdot)$ (Equation 3.30). Let the parameters to π_s be given as $\mathbf{p} = (x,\, y,\, z)^\top$. Then the Jacobian is the matrix of the partial derivatives of the elements of $\pi_s(\mathbf{p})$ by the elements of $\mathbf{p}$. This is easily worked out using standard calculus as

$$\frac{\partial \pi_s(\mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} \frac{f_x}{z} & 0 & -\frac{f_x x}{z^2} \\ 0 & \frac{f_y}{z} & -\frac{f_y y}{z^2} \\ 0 & 0 & -\frac{f_y b}{z^2} \end{bmatrix}. \tag{A.7}$$

When we plug in $\mathbf{p} = \mathbf{y}_i^{\mathcal{C}}$, we obtain the first factor of the product A.6. The derivative of the projection function $\pi_s(\cdot)$ is a building block that will frequently be useful in other Jacobians. Several such basic building blocks will be presented in the next section.

By repeated application of the chain rule the second factor of Equation A.6 can be further decomposed as

$$\frac{\partial \mathbf{y}_i^{\mathcal{C}}}{\partial \mathbf{y}_i} = \frac{\partial \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathbf{y}_i - \mathbf{r}^{\mathcal{WC}}\right)}{\partial \mathbf{y}_i} = \frac{\partial \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\left(\mathbf{y}_i - \mathbf{r}^{\mathcal{WC}}\right)}{\partial\left(\mathbf{y}_i - \mathbf{r}^{\mathcal{WC}}\right)} \cdot \frac{\partial\left(\mathbf{y}_i - \mathbf{r}^{\mathcal{WC}}\right)}{\partial \mathbf{r}^{\mathcal{WC}}} = \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1} \cdot (-\mathbf{I}). \tag{A.8}$$

In summary, we obtain

$$\frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}} = -\frac{\partial \pi_s\left(\mathbf{y}_i^{\mathcal{C}}\right)}{\partial \mathbf{y}_i^{\mathcal{C}}} \cdot \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}. \tag{A.9}$$

The remaining derivatives $\frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{q}^{\mathcal{WC}}}$ and $\frac{\partial \mathbf{y}_i^{\mathcal{I}}}{\partial \mathbf{y}_i}$ can be decomposed analogously.

A.2. Jacobians of Basic Operations

A.2.1. Projection Functions

Stereo Projection from Inverse Depth and Ray

In Section 6.5.1 we introduced the modified stereo projection function $\pi_\rho(\rho, \mathbf{x})$. Its arguments represent the point to be projected as a ray $\mathbf{x} = (x,\, y,\, z)^\top$ and the inverse depth ρ along that ray. π_ρ was defined as follows

$$\pi_\rho(\rho, \mathbf{x}) = \begin{pmatrix} f_x \frac{x}{z} + u_0 \\ f_y \frac{y}{z} + v_0 \\ \rho f_x \frac{b}{z} \end{pmatrix}.$$

The Jacobians of $\pi_\rho(\rho, \mathbf{x})$ by ρ and $\mathbf{x}$ are easily derived as

$$\frac{\partial \pi_\rho(\rho, \mathbf{x})}{\partial \rho} = \begin{bmatrix} 0 \\ 0 \\ f_x \frac{b}{z} \end{bmatrix} \tag{A.10}$$

and

$$\frac{\partial \pi_\rho(\rho, \mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{f_x}{z} & 0 & -f_x \frac{x}{z^2} \\ 0 & \frac{f_y}{z} & -f_y \frac{y}{z^2} \\ 0 & 0 & -\rho f_x \frac{b}{z^2} \end{bmatrix}. \tag{A.11}$$

Projecting and Un-projecting Homogeneous Coordinates

In Section 3.4 the function h to project homogeneous to Euclidean coordinates was defined as

$$h : \mathbb{R}^3 \to \mathbb{R}^2 : \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix} \text{ for } z \neq 0.$$

The Jacobian of the projection of $\mathbf{x} = (x,\, y,\, z)^\top$ by $\mathbf{x}$ is

$$\frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} = \frac{1}{z} \cdot \begin{bmatrix} 1 & 0 & -\frac{x}{z} \\ 0 & 1 & -\frac{y}{z} \end{bmatrix}. \tag{A.12}$$

The un-projection function h^{-1} computes a homogeneous representation of its argument by augmenting a 1 as the third coordinate

$$h^{-1} : \mathbb{R}^2 \to \mathbb{R}^3 : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

The Jacobian of the un-projection of $\mathbf{x} = (x,\, y)^\top$ by $\mathbf{x}$ is

$$\frac{\partial h^{-1}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{A.13}$$

A.2.2. Quaternion Rotations

Quaternion multiplication

The product of two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ was defined in Equation 3.7. Writing out all components explicitly the product is

$$\mathbf{q}_1 \circ \mathbf{q}_2 = \begin{pmatrix} w_1 \\ x_1 \\ y_1 \\ z_1 \end{pmatrix} \circ \begin{pmatrix} w_2 \\ x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ w_1 x_2 + w_2 x_1 + y_1 z_2 - z_1 y_2 \\ w_1 y_2 + w_2 y_1 + z_1 x_2 - x_1 z_2 \\ w_1 z_2 + w_2 z_1 + x_1 y_2 - y_1 x_2 \end{pmatrix}. \tag{A.14}$$

The partial derivatives by the components of the quaternion factors can be easily read off from the above result. For instance, from the first component of the resulting quaternion $w = w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2$ we obtain $\frac{\partial w}{\partial w_1} = w_2$, $\frac{\partial w}{\partial x_1} = -x_2$, etc. Using the partial derivatives, we obtain the Jacobian matrices by the first and second factor,

$$\frac{\partial (\mathbf{q}_1 \circ \mathbf{q}_2)}{\partial \mathbf{q}_1} = \begin{bmatrix} w_2 & -x_2 & -y_2 & -z_2 \\ x_2 & w_2 & z_2 & -y_2 \\ y_2 & -z_2 & w_2 & x_2 \\ z_2 & y_2 & -x_2 & w_2 \end{bmatrix} \tag{A.15}$$

and

$$\frac{\partial (\mathbf{q}_1 \circ \mathbf{q}_2)}{\partial \mathbf{q}_2} = \begin{bmatrix} w_1 & -x_1 & -y_1 & -z_1 \\ x_1 & w_1 & -z_1 & y_1 \\ y_1 & z_1 & w_1 & -x_1 \\ z_1 & -y_1 & x_1 & w_1 \end{bmatrix}. \tag{A.16}$$

Inverse of a Quaternion

The inverse of a quaternion $\mathbf{q} = (q_w, q_x, q_y, q_z)^\top$ is represents the rotation that "undoes" the rotation $\mathbf{q}$. It is obtained by inverting the vector part of $\mathbf{q}$, i.e., negates the axis of rotation, cf. Equation 3.8.

$$\mathbf{q}^{-1} = \begin{pmatrix} q_w \\ -q_x \\ -q_y \\ -q_z \end{pmatrix}. \tag{A.17}$$

The Jacobian of quaternion inversion is

$$\frac{\partial \mathbf{q}^{-1}}{\partial \mathbf{q}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \tag{A.18}$$

Rotation of a Vector

To derive this Jacobian, we express the first convert the quaternion to a rotation matrix and carry out the rotation in matrix form. The rotation matrix $\mathsf{R}_{\mathbf{q}}$ corresponding to the quaternion $\mathbf{q} = (q_w, q_x, q_y, q_z)^\top$ is given in Equation 3.10. Consider the vector $\mathbf{x} = (a, b, c)^\top$. The rotated vector is obtained as

$$\mathsf{R}_{\mathbf{q}}\mathbf{x} = \begin{pmatrix} (q_w^2 + q_x^2 - q_y^2 - q_z^2) \cdot a & + & 2(q_x q_y - q_w q_z) \cdot b & + & 2(q_x q_z + q_w q_y) \cdot c \\ 2(q_x q_y + q_w q_z) \cdot a & + & (q_w^2 - q_x^2 + q_y^2 - q_z^2) \cdot b & + & 2(q_y q_z - q_w q_x) \cdot c \\ 2(q_x q_z - q_w q_y) \cdot a & + & 2(q_y q_z + q_w q_x) \cdot b & + & (q_w^2 - q_x^2 - q_y^2 + q_z^2) \cdot c \end{pmatrix} \tag{A.19}$$

Forming the partial derivatives by the quaternion components, we obtain the Jacobian

$$\frac{\partial \mathsf{R}_{\mathbf{q}}\mathbf{x}}{\partial \mathbf{q}} = 2 \cdot \begin{bmatrix} q_w a - q_z b + q_y c & q_x a + q_y b + q_z c & -q_y a + q_x b + q_w c & -q_z a - q_w b + q_x c \\ q_z a + q_w b - q_x c & q_y a - q_x b - q_w c & q_x a + q_y b + q_z c & q_w a - q_z b + q_y c \\ -q_y a + q_x b + q_w c & q_z a + q_w b - q_x c & -q_w a + q_z b - q_y c & q_x a + q_y b + q_z c \end{bmatrix}. \tag{A.20}$$

Log Map: Converting Quaternion to Exponential Coordinates

Let $\mathbf{q} = (q_w, q_x, q_y, q_z)^\top$ be a quaternion, and let $\boldsymbol{\omega} = \log(\mathbf{q}) = (\omega_x, \omega_y, \omega_z)^\top$ be the corresponding exponential coordinates. The log map was defined in Equation 3.16, which we restate here a bit more verbose (for the case $\mathbf{q} \neq (1, 0, 0, 0)$):

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \log \begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} = \frac{2\cos^{-1}(q_w)}{\sin(\cos^{-1}(q_w))} \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix}. \tag{A.21}$$

We derive the 3×4 Jacobian matrix

$$\frac{\partial \log(\mathbf{q})}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial \omega_x}{\partial q_w} & \frac{\partial \omega_x}{\partial q_x} & \frac{\partial \omega_x}{\partial q_y} & \frac{\partial \omega_x}{\partial q_z} \\ \frac{\partial \omega_y}{\partial q_w} & \frac{\partial \omega_y}{\partial q_x} & \frac{\partial \omega_y}{\partial q_y} & \frac{\partial \omega_y}{\partial q_z} \\ \frac{\partial \omega_z}{\partial q_w} & \frac{\partial \omega_z}{\partial q_x} & \frac{\partial \omega_z}{\partial q_y} & \frac{\partial \omega_z}{\partial q_z} \end{bmatrix} \tag{A.22}$$

of the partial derivatives of exponential components by quaternion components. The partial derivatives are of the form

$$\frac{\partial \omega_i}{\partial q_w} = 2q_i \underbrace{\left(\frac{q_w \cos^{-1}(q_w)}{(1 - q_w^2)^{\frac{3}{2}}} - \frac{1}{1 - q_w^2} \right)}_{\mathbf{k}} \tag{A.23}$$

$$\frac{\partial \omega_i}{\partial q_i} = \underbrace{2\frac{\cos^{-1}(q_w)}{\sin(\cos^{-1}(q_w))}}_{\mathbf{p}} \tag{A.24}$$

$$\frac{\partial \omega_i}{\partial q_j} = 0 \tag{A.25}$$

for $i, j \in \{x, y, z\}$ and $i \neq j$.

In the neighbourhood of $q_w \to 1$ we run into numerical problems because $(1 - q_w^2) \to 0$ in (A.23), and $\sin(\cos^{-1}(q_w)) \to 0$ in (A.24). We can avoid this by plugging in Taylor expansions for the marked terms $\mathbf{k}$ and $\mathbf{p}$.

We can write the term $\mathbf{k}$ as

$$\mathbf{k} = \frac{(1-x)\cos^{-1}(1-x)}{(1-(1-x)^2)^{\frac{3}{2}}} - \frac{1}{1-(1-x)^2} \quad \text{with } x = 1 - q_w. \tag{A.26}$$

Forming the Taylor expansion at $x = 0$ we obtain

$$\mathbf{k} = \frac{(1-x)\cos^{-1}(1-x)}{(1-(1-x)^2)^{\frac{3}{2}}} - \frac{1}{1-(1-x)^2} = -\frac{1}{3} - \frac{4x}{15} - \frac{6x^2}{35} - \frac{32x^3}{315} - \dots \tag{A.27}$$

Similarly, we can write the $\mathbf{p}$ as

$$\mathbf{p} = 2\frac{\cos^{-1}(1-x)}{\sin\left(\cos^{-1}(1-x)\right)} \quad \text{with } x = 1 - q_w. \tag{A.28}$$

Forming the Taylor expansion at $x = 0$ we obtain

$$\frac{2\cos^{-1}(1-x)}{\sin\left(\cos^{-1}(1-x)\right)} = 2 + \frac{2x}{3} + \frac{4x^2}{15} + \frac{4x^3}{35} + \dots \tag{A.29}$$

After discarding higher-order terms we obtain the following approximation for the derivatives

$$\frac{\partial \omega_i}{\partial q_w} = 2q_i \left(-\frac{1}{3} - \frac{4(1-q_w)}{15} - \frac{6(1-q_w)^2}{35} \right) \tag{A.30}$$

$$\frac{\partial \omega_i}{\partial q_i} = 2 + \frac{2(1-q_w)}{3} + \frac{4(1-q_w)^2}{15} \tag{A.31}$$

$$\frac{\partial \omega_i}{\partial q_j} = 0. \tag{A.32}$$

These approximations are used instead of (A.23)-(A.25) in the neighbourhood of $q_w \to 1$.

A.2.3. Exponential Rotations

Exponential Map: Converting Exponential Coordinates to Quaternion

Let $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^\top$ be a rotation in exponential coordinates, and let $\mathbf{q} = \exp(\boldsymbol{\omega}) = (q_w, q_x, q_y, q_z)^\top$ be the corresponding quaternion. The exp map was defined in Equation 3.13, which we restate here a bit more verbose (for the case $\boldsymbol{\omega} \neq (0,0,0)$):

$$\begin{pmatrix} q_w \\ q_x \\ q_y \\ q_z \end{pmatrix} = \exp \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{1}{2}\theta\right) \\ \frac{\sin\left(\frac{1}{2}\theta\right)}{\theta}\omega_x \\ \frac{\sin\left(\frac{1}{2}\theta\right)}{\theta}\omega_y \\ \frac{\sin\left(\frac{1}{2}\theta\right)}{\theta}\omega_z \end{pmatrix} \quad \text{with } \theta = \sqrt{\omega_x^2 + \omega_y^2 + \omega_z^2} \tag{A.33}$$

We derive the 4×3 Jacobian matrix

$$\frac{\partial \exp(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \begin{pmatrix} \frac{\partial q_w}{\partial \omega_x} & \frac{\partial q_w}{\partial \omega_y} & \frac{\partial q_w}{\partial \omega_z} \\ \frac{\partial q_x}{\partial \omega_x} & \frac{\partial q_x}{\partial \omega_y} & \frac{\partial q_x}{\partial \omega_z} \\ \frac{\partial q_y}{\partial \omega_x} & \frac{\partial q_y}{\partial \omega_y} & \frac{\partial q_y}{\partial \omega_z} \\ \frac{\partial q_z}{\partial \omega_x} & \frac{\partial q_z}{\partial \omega_y} & \frac{\partial q_z}{\partial \omega_z} \end{pmatrix} \tag{A.34}$$

of the partial derivatives of quaternion components by exponential components. The partial derivatives are of the form

$$\frac{\partial q_w}{\partial \omega_i} = -\frac{1}{2}\omega_i \frac{\sin(\frac{1}{2}\theta)}{\theta} \tag{A.35}$$

$$\frac{\partial q_i}{\partial \omega_i} = \frac{1}{2}\omega_i^2 \frac{\cos(\frac{1}{2}\theta)}{\theta^2} - \omega_i^2 \frac{\sin(\frac{1}{2}\theta)}{\theta^3} + \frac{\sin(\frac{1}{2}\theta)}{\theta} \tag{A.36}$$

$$\frac{\partial q_j}{\partial \omega_i} = \frac{1}{2}\omega_j \omega_i \frac{\cos(\frac{1}{2}\theta)}{\theta^2} - \omega_j \omega_i \frac{\sin(\frac{1}{2}\theta)}{\theta^3} \tag{A.37}$$

where $i, j \in \{x, y, z\}$ and $i \neq j$.

In the neighbourhood of $\theta \to 0$ we plug in the Taylor expansions of sine and cosine. After simplifying and discarding higher-order terms we obtain

$$\frac{\partial q_w}{\partial \omega_i} = -\frac{1}{2}\omega_i \left(\frac{1}{2} - \frac{\theta^2}{48}\right) \tag{A.38}$$

$$\frac{\partial q_i}{\partial \omega_i} = \frac{\omega_i^2}{24}\left(\frac{\theta^2}{40} - 1\right) + \left(\frac{1}{2} - \frac{\theta^2}{48}\right) \tag{A.39}$$

$$\frac{\partial q_j}{\partial \omega_i} = \frac{\omega_j \omega_i}{24}\left(\frac{\theta^2}{40} - 1\right) \tag{A.40}$$

For details, we refer to Grassia (1998).

Rotation of a Vector

Consider the rotation a vector $\mathbf{x} = (a, b, c)^\top$ by an exponential vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^\top$. We can obtain the rotated vectori $\mathbf{x}'$ by converting the exponential to a quaternion, then converting the quaternion to a rotation matrix, and finally multiplying the rotation matrix by the vector.

$$\mathbf{x}' = \mathsf{R}_{\boldsymbol{\omega}} = \mathsf{R}_{\exp(\boldsymbol{\omega})}\mathbf{x}. \tag{A.41}$$

The Jacobian is obtained via the chain rule

$$\frac{\partial \mathsf{R}_{\boldsymbol{\omega}}\mathbf{x}}{\partial \boldsymbol{\omega}} = \frac{\partial \mathsf{R}_{\exp(\boldsymbol{\omega})}\mathbf{x}}{\partial \exp(\boldsymbol{\omega})} \cdot \frac{\partial \exp(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}, \tag{A.42}$$

where the first factor is the quaternion's "rotation of a vector" Jacobian (Equation A.20), and the second factor is the Jacobian of the exp map given above (Equation A.34).

A.2.4. Polar Coordinates

Azimuth-Elevation to Unit Vector

The unit vector corresponding to polar coordinates $\boldsymbol{\theta} = (\theta_x, \theta_y)^\top$ is computed as:

$$\mathbf{n}(\boldsymbol{\theta}) = \begin{pmatrix} \sin(\theta_x)\cos(\theta_y) \\ \sin(\theta_y) \\ \cos(\theta_x)\cos(\theta_y) \end{pmatrix} \tag{A.43}$$

The Jacobian of the unit vector by the polar coordinates is

$$\frac{\partial \mathbf{n}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \cos(\theta_x)\cos(\theta_y) & -\sin(\theta_x)\sin(\theta_y) \\ 0 & \cos(\theta_y) \\ -\sin(\theta_x)\cos(\theta_y) & -\cos(\theta_x)\sin(\theta_y) \end{bmatrix}. \tag{A.44}$$

A.3. Jacobians of the Constant-velocity Process Model

In this section we derive the Jacobians for the constant-velocity motion model introduced in Section 4.5. We need the partial derivatives of Equation 4.12 by the state at the previous time-step, $\mathbf{x}_{t-1}$, and by the process noise $\boldsymbol{\varepsilon}_t$. The derivatives are evaluated at $\mathbf{x}_{t-1} = \boldsymbol{\mu}_{t-1},\ \boldsymbol{\varepsilon}_t = \mathbf{0}$.

Let us first consider the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{t-1}}$. The map part of the state vector is static, the corresponding Jacobian block is the identity matrix. Only the camera state, $\mathbf{x}_v$, is modified by the process model. Moreover, its evolution independent of the map. Thus, the block layout of the full Jacobian is

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{t-1}} = \begin{bmatrix} \frac{\partial \mathbf{x}_{v;t}}{\partial \mathbf{x}_{v;t-1}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{A.45}$$

The process model acts on the components of $\mathbf{x}_v$ as follows, cf. Equation 4.12.

$$\mathbf{r}_t^{\mathcal{WC}} = \mathbf{r}_{t-1}^{\mathcal{WC}} + \Delta t \cdot \mathbf{v}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2} \cdot \mathbf{a}_t \tag{A.46}$$

$$\mathbf{q}_t^{\mathcal{WC}} = \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2} \cdot \boldsymbol{\alpha}_t\right) \circ \mathbf{q}_{t-1}^{\mathcal{WC}} \tag{A.47}$$

$$\mathbf{v}_t^{\mathcal{W}} = \mathbf{v}_{t-1}^{\mathcal{W}} + \Delta t \cdot \mathbf{a}_t \tag{A.48}$$

$$\boldsymbol{\omega}_t^{\mathcal{W}} = \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \Delta t \cdot \boldsymbol{\alpha}_t \tag{A.49}$$

The camera state Jacobian is of the form

$$\frac{\partial \mathbf{x}_{v;t}}{\partial \mathbf{x}_{v;t-1}} = \begin{bmatrix} \frac{\partial \mathbf{r}_t^{\mathcal{WC}}}{\partial \mathbf{r}_{t-1}^{\mathcal{WC}}} & \mathbf{0} & \frac{\partial \mathbf{r}_t^{\mathcal{WC}}}{\partial \mathbf{v}_{t-1}^{\mathcal{W}}} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \mathbf{q}_{t-1}^{\mathcal{WC}}} & \mathbf{0} & \frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \boldsymbol{\omega}_{t-1}^{\mathcal{W}}} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{v}_t^{\mathcal{W}}}{\partial \mathbf{v}_{t-1}^{\mathcal{W}}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \boldsymbol{\omega}_t^{\mathcal{W}}}{\partial \boldsymbol{\omega}_{t-1}^{\mathcal{W}}} \end{bmatrix} \tag{A.50}$$

The non-zero blocks are obtained by differentiating (A.46)-(A.49). The derivatives are trivial, aside from those involving $\mathbf{q}^{\mathcal{WC}}$. We have

$$\frac{\partial \mathbf{r}_t^{\mathcal{WC}}}{\partial \mathbf{r}_{t-1}^{\mathcal{WC}}} = \mathbf{I} \tag{A.51}$$

$$\frac{\partial \mathbf{r}_t^{\mathcal{WC}}}{\partial \mathbf{v}_{t-1}^{\mathcal{W}}} = \Delta t \cdot \mathbf{I} \tag{A.52}$$

$$\frac{\partial \mathbf{v}_t^{\mathcal{W}}}{\partial \mathbf{v}_{t-1}^{\mathcal{W}}} = \mathbf{I} \tag{A.53}$$

$$\frac{\partial \boldsymbol{\omega}_t^{\mathcal{W}}}{\partial \boldsymbol{\omega}_{t-1}^{\mathcal{W}}} = \mathbf{I} \tag{A.54}$$

For the Jacobians of the quaternion (Equation A.47) we get the following. Remember that the Jacobian is evaluated at $\boldsymbol{\varepsilon}_t = \mathbf{0}$, so the $\boldsymbol{\alpha}_t$ term can be dropped. For the derivative by $\mathbf{q}_{t-1}^{\mathcal{WC}}$ we have

$$\frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \mathbf{q}_{t-1}^{\mathcal{WC}}} = \frac{\partial \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}}\right) \circ \mathbf{q}_{t-1}^{\mathcal{WC}}}{\partial \mathbf{q}_{t-1}^{\mathcal{WC}}}, \tag{A.55}$$

which is the "quaternion product by second argument" Jacobian (Equation A.16).

For the derivative by $\boldsymbol{\omega}^{\mathcal{W}}$ we obtain

$$\frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \boldsymbol{\omega}_{t-1}^{\mathcal{W}}} = \frac{\partial \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}}\right) \circ \mathbf{q}_{t-1}^{\mathcal{WC}}}{\partial \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}}\right)} \cdot \frac{\partial \exp\left(\Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}}\right)}{\partial \Delta t \cdot \boldsymbol{\omega}_{t-1}^{\mathcal{W}}} \cdot \Delta t \cdot \mathbf{I} \tag{A.56}$$

where the first factor is the "quaternion product by first argument" Jacobian (Equation A.15) and the second factor is the "exponential map" Jacobian (Equation A.34).

Let us now consider the Jacobian $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}_t}$, the partial derivatives by the process noise. Again, we note that the map state is static, thus the full Jacobian is

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\varepsilon}_t} = \begin{bmatrix} \frac{\partial \mathbf{x}_{v;t}}{\partial \boldsymbol{\varepsilon}_t} \\ \mathbf{0} \end{bmatrix} \tag{A.57}$$

For the camera state block, we have

$$\frac{\partial \mathbf{x}_{v;t}}{\partial \boldsymbol{\varepsilon}_t} = \begin{bmatrix} \frac{\partial \mathbf{r}_t^{\mathcal{WC}}}{\partial \mathbf{a}_t} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \boldsymbol{\alpha}_t} \\ \frac{\partial \mathbf{v}_t^{\mathcal{W}}}{\partial \mathbf{a}_t} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \boldsymbol{\omega}_t^{\mathcal{W}}}{\partial \boldsymbol{\alpha}_t} \end{bmatrix}. \tag{A.58}$$

Aside from the $\frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \boldsymbol{\alpha}_t}$, the derivatives are again trivial.

$$\frac{\partial \mathbf{r}_t^{\mathcal{WC}}}{\partial \mathbf{a}_t} = \frac{\Delta t^2}{2} \cdot \mathbf{I} \tag{A.59}$$

$$\frac{\partial \mathbf{v}_t^{\mathcal{W}}}{\partial \mathbf{a}_t} = \Delta t \cdot \mathbf{I} \tag{A.60}$$

$$\frac{\partial \boldsymbol{\omega}_t^{\mathcal{W}}}{\partial \boldsymbol{\alpha}_t} = \Delta t \cdot \mathbf{I} \tag{A.61}$$

Finally, for the Jacobian of the quaternion (Equation A.47) we obtain

$$\frac{\partial \mathbf{q}_t^{\mathcal{WC}}}{\partial \boldsymbol{\alpha}_t} = \frac{\partial \exp\left(\Delta t\, \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2}\, \boldsymbol{\alpha}_t\right) \circ \mathbf{q}_{t-1}^{\mathcal{WC}}}{\partial \exp\left(\Delta t\, \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2}\, \boldsymbol{\alpha}_t\right)} \cdot \frac{\partial \exp\left(\Delta t\, \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2}\, \boldsymbol{\alpha}_t\right)}{\partial \Delta t\, \boldsymbol{\omega}_{t-1}^{\mathcal{W}} + \frac{\Delta t^2}{2}\, \boldsymbol{\alpha}_t} \cdot \frac{\Delta t^2}{2} \cdot \mathbf{I}. \tag{A.62}$$

Again, we use the Jacobians for "quaternion product by first argument" (Equation A.15) and "exponential map" (Equation A.34).

A.4. Jacobians of the View-Point Based Measurement Model

In this section we derive the Jacobians for the view-point based measurement model introduced in Section 6.5.1. We need the derivatives by the noise $\boldsymbol{\delta}$ and state vector $\mathbf{x}$. We restrict ourselves to the measurement of a single feature. In the same way as the measurement model, the Jacobians for the case of measuring multiple features can be obtained simply by stacking single-feature Jacobians.

So lets consider the measurement of $\mathbf{y} = (\mathbf{c},\, \boldsymbol{\phi},\, \rho)^\top$. For convenience we restate the measurement Equation (6.23)

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{y}^{\mathcal{I}} + \boldsymbol{\delta} = \pi_\rho\big(\rho\,,\, \underbrace{\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}} \underbrace{\big(\mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}} + \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)\big)}_{\mathbf{k}}}_{\mathbf{p}}\big) + \boldsymbol{\delta}. \tag{A.63}$$

As indicated, we introduce the following abbreviations

$$\mathbf{k} = \mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}} + \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right) \tag{A.64}$$

$$\mathbf{p} = \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathbf{k}. \tag{A.65}$$

Using these abbreviations we write the measurement equation as

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{y}^{\mathcal{I}} + \boldsymbol{\delta} = \pi_\rho\left(\rho,\, \mathbf{p}\right) + \boldsymbol{\delta}. \tag{A.66}$$

We have additive measurement noise which immediately gives $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}} = \mathbf{I}$.

For $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ we note that the measurement function depends only on the $\mathbf{r}^{\mathcal{WC}}$, $\mathbf{q}^{\mathcal{WC}}$ and $\mathbf{y}$ components of the state which results in the following sparse Jacobian:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{x}} = \left[\begin{array}{cccccc} \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}} & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{q}^{\mathcal{WC}}} & \mathbf{0} & \cdots & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{y}} & \mathbf{0} \quad \cdots \end{array}\right]. \tag{A.67}$$

Each non-zero block in turn can be decomposed using the chain rule. The basic derivatives that have been introduced in the previous section will be put to use now. We will indicate after each derivative the basic derivatives that are used.

For the derivative by the camera translation we obtain

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}} = \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \frac{\partial \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathbf{k}}{\partial \mathbf{k}} \cdot \frac{\partial \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)}{\partial \mathbf{r}^{\mathcal{WC}}} \tag{A.68}$$

$$= \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}} \cdot (-\rho\mathbf{I}) \tag{A.69}$$

$$= \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot (-\rho\mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}) \tag{A.70}$$

The first factor is the partial derivative of the stereo projection by the ray $\mathbf{p}$ which is given in Equation (A.11)

For the derivative by the camera rotation we have

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{q}^{\mathcal{WC}}} = \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \frac{\partial \mathsf{R}^{-1}_{\mathbf{q}^{\mathcal{WC}}}\mathbf{k}}{\partial \mathbf{q}^{\mathcal{WC}-1}} \cdot \frac{\partial \mathbf{q}^{\mathcal{WC}-1}}{\partial \mathbf{q}^{\mathcal{WC}}} \tag{A.71}$$

The first factor again is "stereo projection by ray" (Equation A.11). For the second and third factor we need the Jacobians for "rotated vector by quaternion rotation" (Equation A.20) and "inverted quaternion by quaternion" (Equation A.18).

The third non-zero block in (A.67) is the derivative by the feature vector. We further decompose this and consider three blocks: the derivatives by the anchor pose translation, rotation, and by the inverse depth

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{c}} & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \rho} \end{bmatrix}. \tag{A.72}$$

The derivative by the anchor translation is completely analogous to the derivative by the current camera translation (A.70).

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{c}} = \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \frac{\partial \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathbf{k}}{\partial \mathbf{k}} \cdot \frac{\partial \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)}{\partial \mathbf{c}} \tag{A.73}$$

$$= \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \left(\rho \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\right). \tag{A.74}$$

The derivative by the anchor rotation is

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \boldsymbol{\phi}} = \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \frac{\partial \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathbf{k}}{\partial \mathbf{k}} \cdot \frac{\partial \mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}}}{\partial \boldsymbol{\phi}} \tag{A.75}$$

$$= \frac{\partial \pi_\rho\left(\rho,\, \mathbf{p}\right)}{\partial \mathbf{p}} \cdot \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1} \cdot \frac{\partial \mathsf{R}_{\boldsymbol{\phi}}\mathbf{m}^{\mathcal{A}}}{\partial \boldsymbol{\phi}}, \tag{A.76}$$

using "rotated vector by exponential rotation" (Equation A.42) in the last factor.

The derivative by ρ is less obvious. Both arguments of the projection function π_ρ depend on ρ. However, if we stack the arguments of π_ρ in a vector $(\rho,\, \mathbf{p})^\top$ we can also apply the chain rule in this case

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \rho} = \frac{\partial \pi_\rho\left(\rho, \mathbf{p}\right)}{\partial \rho} \tag{A.77}$$

$$= \frac{\partial \pi_\rho\left(\rho, \mathbf{p}\right)}{\partial \binom{\rho}{\mathbf{p}}} \cdot \frac{\partial \binom{\rho}{\mathbf{p}}}{\partial \rho} \tag{A.78}$$

$$= \begin{bmatrix} \frac{\partial \pi_\rho(\rho, \mathbf{x})}{\partial \rho}\Big|_{\mathbf{x}=\mathbf{p}} & \frac{\partial \pi_\rho(x, \mathbf{p})}{\partial \mathbf{p}}\Big|_{x=\rho} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \rho}{\partial \rho} \\ \frac{\partial \mathbf{p}}{\partial \rho} \end{bmatrix} \tag{A.79}$$

The left block in the first factor is the Jacobian of "stereo projection by inverse depth" (Equation A.10). For the second factor, we have $\frac{\partial \rho}{\partial \rho} = 1$ and

$$\frac{\partial \mathbf{p}}{\partial \rho} = \frac{\partial \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathbf{k}}{\partial \mathbf{k}} \cdot \frac{\partial \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right)}{\partial \rho} = \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1} \cdot \left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right). \tag{A.80}$$

Plugging these into the matrix product (A.79) we obtain

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \rho} = \frac{\partial \pi_\rho\left(\rho,\, \mathbf{x}\right)}{\partial \rho}\Big|_{\mathbf{x}=\mathbf{p}} + \frac{\partial \pi_\rho\left(x,\, \mathbf{p}\right)}{\partial \mathbf{p}}\Big|_{x=\rho} \cdot \mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1} \cdot \left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right) \tag{A.81}$$

This completes the derivation of the view-point based measurement Jacobians.

A.5. Jacobians of the View-Point Based Feature Initialisation

For augmenting the state with a new view-point based feature we need the derivatives of the initialisation function (Equation 6.34, restated for convenience here)

$$\mathbf{y} = \mathbf{g}(d, \mathbf{x}) = \begin{pmatrix} \mathbf{c} \\ \boldsymbol{\phi} \\ \rho \end{pmatrix} = \begin{pmatrix} \mathbf{r}^{\mathcal{WC}} \\ \log\left(\mathbf{q}^{\mathcal{WC}}\right) \\ d\frac{z_m}{f_x b} \end{pmatrix}$$

by the initially measured disparity d and the state vector $\mathbf{x}$. Let us first note that the initial observation pose $\mathbf{c}$, $\boldsymbol{\phi}$ is independent of the disparity. Furthermore, it is a copy of the current camera pose $\mathbf{r}^{\mathcal{WC}}$, $\mathbf{q}^{\mathcal{WC}}$. Likewise, the inverse depth ρ is independent of the current state. Furthermore, it is a linear function of d. This makes the Jacobians particularly simple: For the derivative by d we obtain

$$\frac{\partial \mathbf{g}}{\partial d} = \begin{bmatrix} 0 \\ 0 \\ \frac{\partial \rho}{\partial d} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{z_m}{f_x b} \end{bmatrix}. \tag{A.82}$$

For the derivative by the state we obtain

$$\frac{\partial \mathbf{g}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{r}^{\mathcal{WC}}} & \frac{\partial \mathbf{g}}{\partial \mathbf{q}^{\mathcal{WC}}} & \mathbf{0} & \dots \end{bmatrix} \tag{A.83}$$

$$= \begin{bmatrix} \frac{\partial \mathbf{c}}{\partial \mathbf{r}^{\mathcal{WC}}} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{q}^{\mathcal{WC}}} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \end{bmatrix} \tag{A.84}$$

$$= \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\partial \log\left(\mathbf{q}^{\mathcal{WC}}\right)}{\partial \mathbf{q}^{\mathcal{WC}}} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \end{bmatrix}. \tag{A.85}$$

The basic derivative of "log of quaternion by quaternion" (Equation A.22) can be found in Section A.2.

A.6. Jacobians of the Inverse Depth Bundle Measurement Model

In this section we give the Jacobians for the inverse depth bundle measurement model discussed in Section 6.6. We need the derivatives by the noise $\boldsymbol{\delta}$ and state vector $\mathbf{x}$. The bundle model was obtained by splitting the view-point based representation into an anchor part and a feature part. Likewise, the Jacobian is obtained by splitting the view-point based Jacobian discussed in Appendix A.4.

We restrict ourselves to the measurement of a single feature. The Jacobians for the case of measuring multiple features can be obtained simply by stacking single-feature Jacobians. So lets consider the measurement of $\mathbf{y} = \rho$ with respect to anchor $\mathbf{a} = (\mathbf{c}, \boldsymbol{\phi})^\top$.

The measurement function, as before, is

$$\mathbf{z} = \mathbf{h}(\mathbf{x}, \boldsymbol{\delta}) = \mathbf{y}^{\mathcal{I}} + \boldsymbol{\delta} \tag{A.86}$$

cf. Equation (6.23).

We have additive measurement noise, which immediately gives $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\delta}} = \mathbf{I}$.

For $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$ we note that the measurement function depends only on the $\mathbf{r}^{\mathcal{WC}}$, $\mathbf{q}^{\mathcal{WC}}$, as well as the $\mathbf{a}$ and $\mathbf{y}$ components of the state which results in the following sparse Jacobian:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}} & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{q}^{\mathcal{WC}}} & \mathbf{0} & \cdots & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{a}} & \mathbf{0} & \cdots & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{y}} & \mathbf{0} & \cdots \end{bmatrix}. \tag{A.87}$$

The blocks $\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{r}^{\mathcal{WC}}}$ and $\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{q}^{\mathcal{WC}}}$ are the same as before, given in Equations A.70 and A.71, respectively. For $\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{a}}$ and $\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{y}}$, we extract anchor and feature blocks from Equation A.72. We have

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{a}} = \begin{bmatrix} \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{c}} & \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \boldsymbol{\phi}} \end{bmatrix} \tag{A.88}$$

where the blocks are given in Equations A.74 and A.76. Finally,

$$\frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \mathbf{y}} = \frac{\partial \mathbf{y}^{\mathcal{I}}}{\partial \rho} \tag{A.89}$$

is given in Equation A.79.

A.7. Jacobians of the Inverse Depth Bundle Feature Initialisation

To augment a new anchor into the state, we need the derivative of the anchor initialisation function (Equation 6.38, restated for convenience here)

$$\mathbf{a} = \mathbf{f}(\mathbf{x}) = \begin{pmatrix} \mathbf{c} \\ \boldsymbol{\phi} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^{\mathcal{WC}} \\ \log\left(\mathbf{q}^{\mathcal{WC}}\right) \end{pmatrix}. \tag{A.90}$$

The function is identical to the first two rows of the view-point based initialisation. Likewise, the Jacobian by the state comprises the first two rows of Equation A.85

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \frac{\partial \log\left(\mathbf{q}^{\mathcal{WC}}\right)}{\partial \mathbf{q}^{\mathcal{WC}}} & \mathbf{0} & \dots \end{bmatrix}. \tag{A.91}$$

To add a new feature with respect to the anchor, we need the derivative of the feature initialisation function (Equation 6.40, restated for convenience here)

$$\mathbf{y} = \rho = \mathbf{g}(d) = d\frac{z_m}{f_x b} \tag{A.92}$$

The function is identical to the last row of the view-point based initialisation. Likewise, the Jacobian by d comprises the last row of Equation A.82

$$\frac{\partial \mathbf{g}}{\partial d} = \frac{z_m}{f_x b}. \tag{A.93}$$

A.8. Jacobians of the Planar Measurement Model

Before we look at the Jacobians of the planar feature model, we require some basic derivatives of operations with stacked matrices.

A.8.1. Jacobians of Operations with Stacked Matrices

In Section 7.4.3, we have defined functions to stack and unstack 3×3 matrix into a vector, which we repeat for convenience here

$$S(\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}) = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{pmatrix} \qquad S^{-1}(\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \\ a_9 \end{pmatrix}) = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix}. \tag{A.94}$$

In computing the derivatives of the warp function, the stacking is "pushed inside" by the application of the chain rule, as we will see shortly. We will need Jacobians of some basic operations on stacked 3×3 matrices.

Stacked Matrix Product

The first of these operations is the product of two matrices A and B stacked into a vector $S(\mathsf{AB})$. We need the derivative of this stacked product by the stacked first factor $S(\mathsf{A})$ and the stacked second factor $S(\mathsf{B})$.

Let A and B denote the matrices

$$\mathsf{A} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \qquad \text{and} \qquad \mathsf{B} = \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix}. \tag{A.95}$$

Stacking the result of AB gives

$$S(\mathsf{AB}) = \begin{pmatrix} a_1b_1 + a_2b_4 + a_3b_7 \\ a_1b_2 + a_2b_5 + a_3b_8 \\ a_1b_3 + a_2b_6 + a_3b_9 \\ a_4b_1 + a_5b_4 + a_6b_7 \\ a_4b_2 + a_5b_5 + a_6b_8 \\ a_4b_3 + a_5b_6 + a_6b_9 \\ a_7b_1 + a_8b_4 + a_9b_7 \\ a_7b_2 + a_8b_5 + a_9b_8 \\ a_7b_3 + a_8b_6 + a_9b_9 \end{pmatrix}. \tag{A.96}$$

We obtain the following Jacobians by the stacked first and second factor. For better readability we have left 0 entries empty.

$$\frac{\partial S\left(\mathsf{AB}\right)}{\partial S\left(\mathsf{A}\right)} = \begin{bmatrix} b_1 & b_4 & b_7 & & & & & & \\ b_2 & b_5 & b_8 & & & & & & \\ b_3 & b_6 & b_9 & & & & & & \\ & & & b_1 & b_4 & b_7 & & & \\ & & & b_2 & b_5 & b_8 & & & \\ & & & b_3 & b_6 & b_9 & & & \\ & & & & & & b_1 & b_4 & b_7 \\ & & & & & & b_2 & b_5 & b_8 \\ & & & & & & b_3 & b_6 & b_9 \end{bmatrix} = \begin{bmatrix} \mathsf{B}^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathsf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathsf{B}^\top \end{bmatrix} \tag{A.97}$$

$$\frac{\partial S\left(\mathsf{AB}\right)}{\partial S\left(\mathsf{B}\right)} = \begin{bmatrix} a_1 & & & a_2 & & & a_3 & & \\ & a_1 & & & a_2 & & & a_3 & \\ & & a_1 & & & a_2 & & & a_3 \\ a_4 & & & a_5 & & & a_6 & & \\ & a_4 & & & a_5 & & & a_6 & \\ & & a_4 & & & a_5 & & & a_6 \\ a_7 & & & a_8 & & & a_9 & & \\ & a_7 & & & a_8 & & & a_9 & \\ & & a_7 & & & a_8 & & & a_9 \end{bmatrix}. \tag{A.98}$$

Vector Outer Product

Next is the outer product of two 3-vectors (a 3×3 matrix) stacked into a vector. Let $\mathbf{x}$ and $\mathbf{y}$ denote the vectors

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \text{and} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}. \tag{A.99}$$

Stacking their outer product yields

$$S(\mathbf{x}\mathbf{y}^\top) = \begin{pmatrix} x_1 y_1 \\ x_1 y_2 \\ x_1 y_3 \\ x_2 y_1 \\ x_2 y_2 \\ x_2 y_3 \\ x_3 y_1 \\ x_3 y_2 \\ x_3 y_3 \end{pmatrix}. \tag{A.100}$$

We obtain the following Jacobians by the first and second vector.

$$\frac{\partial S(\mathbf{x}\mathbf{y}^\top)}{\partial \mathbf{x}} = \begin{bmatrix} y_1 & & \\ y_2 & & \\ y_3 & & \\ & y_1 & \\ & y_2 & \\ & y_3 & \\ & & y_1 \\ & & y_2 \\ & & y_3 \end{bmatrix} \tag{A.101}$$

$$\frac{\partial S(\mathbf{x}\mathbf{y}^\top)}{\partial \mathbf{y}} = \begin{bmatrix} x_1 & & \\ & x_1 & \\ & & x_1 \\ x_2 & & \\ & x_2 & \\ & & x_2 \\ x_3 & & \\ & x_3 & \\ & & x_3 \end{bmatrix}. \tag{A.102}$$

Rotation matrix from quaternion

We need the Jacobian of the stacked rotation matrix corresponding to a quaternion $\mathbf{q}$ by that quaternion. Consider the unit quaternion $\mathbf{q} = (w, x, y, z)^\top$. The rotation matrix corresponding to $\mathbf{q}$ is

$$\mathsf{R}_\mathbf{q} = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & w^2 - x^2 - y^2 + z^2 \end{bmatrix}. \tag{A.103}$$

We obtain the following derivatives for the stacked rotation matrix.

$$\frac{\partial S\left(\mathsf{R}_\mathbf{q}\right)}{\partial \mathbf{q}} = 2 \cdot \begin{bmatrix} w & x & -y & -z \\ -z & y & x & -w \\ y & z & w & x \\ z & y & x & w \\ w & -x & y & -z \\ -x & -w & z & y \\ -y & z & -w & x \\ x & w & z & y \\ w & -x & -y & z \end{bmatrix}. \tag{A.104}$$

Matrix-Vector Product

Finally, we need the Jacobian of the product of an unstacked 9-vector $\mathbf{p}$ (a 3×3 matrix) and a 3-vector $\mathbf{x}$ by the vector $\mathbf{p}$. Let $\mathbf{p}$ be any 9-vector, and let

$$\mathbf{x} = (x,\, y,\, z)^\top . \tag{A.105}$$

Then we have

$$\frac{\partial S^{-1}(\mathbf{p})\,\mathbf{x}}{\partial \mathbf{p}} = \begin{bmatrix} x & y & z & & & & & & \\ & & & x & y & z & & & \\ & & & & & & x & y & z \end{bmatrix}. \tag{A.106}$$

Again, for better readability we have left 0 entries empty.

A.8.2. Jacobian of a Pixel Measurement

Now, we can derive the Jacobians for the planar feature measurement model introduced in Section 7.4.2. We restrict the examination to the measurement of a single feature. The Jacobians for the case of measuring multiple features can be obtained by stacking single-feature Jacobians. Furthermore, we consider the measurement of a single pixel z_j. For measurement of multiple pixels the single-pixel Jacobians are simply stacked in a matrix.

The measurement model for a pixel $\mathbf{u}_j^{\mathcal{C}}$ is (cf. Equation 7.9)

$$z_j = h_j(\mathbf{x}) + \delta_j \tag{A.107}$$

We need the derivatives by the noise δ_j and the state vector $\mathbf{x}$. The measurement noise is additive. We have $\frac{\partial z_j}{\partial \delta_j} = 1$.

As discussed in Section 7.4.3, the Jacobian of $h_j(\mathbf{x})$ by $\mathbf{x}$ breaks down as follows (cf. Equation 7.17)

$$\frac{\partial h_j(\mathbf{x})}{\partial \mathbf{x}} = \mathsf{H}_{T;j} \cdot \mathsf{H}_w = \underbrace{\nabla T\left(\mathbf{u}_j^{\mathcal{A}}\right) \cdot \frac{\partial w^{\mathsf{H}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{p}\right)}{\partial \mathbf{p}}}_{\mathsf{H}_{T;j}} \cdot \underbrace{\frac{\partial \mathbf{p}\,(\mathbf{x})}{\partial \mathbf{x}}}_{\mathsf{H}_w} \tag{A.108}$$

Jacobian of Intensity by Stacked Homography

Let us look at the first factor, $\mathsf{H}_{T;j}$. The gradient $\nabla T\left(\mathbf{u}_j^{\mathcal{A}}\right)$ is numerically computed from the template image. In general, $\mathbf{u}_j^{\mathcal{A}}$ is not an integer pixel coordinate and the gradient has to be interpolated from nearby pixels. For the derivative of the general warp w^{H} we obtain with the chain rule

$$\frac{\partial w^{\mathsf{H}}\left(\mathbf{u}_j^{\mathcal{C}}\,;\mathbf{p}\right)}{\partial \mathbf{p}} = \frac{\partial h\left(S^{-1}(\mathbf{p}) \cdot h^{-1}\left(\mathbf{u}_j^{\mathcal{C}}\right)\right)}{\partial S^{-1}(\mathbf{p}) \cdot h^{-1}\left(\mathbf{u}_j^{\mathcal{C}}\right)} \cdot \frac{\partial S^{-1}(\mathbf{p}) \cdot h^{-1}\left(\mathbf{u}_j^{\mathcal{C}}\right)}{\partial \mathbf{p}}. \tag{A.109}$$

The first factor is "un-project vector" (Equation A.13), and the second factor is "stacked matrix-vector product" (Equation A.106).

Altogether, we get the following expression for the Jacobian

$$\mathsf{H}_{T;j} = \frac{1}{z} \cdot \nabla T\left(\mathbf{u}_j^{\mathcal{A}}\right) \cdot \begin{bmatrix} u & v & 1 & 0 & 0 & 0 & -u\frac{x}{z} & -v\frac{x}{z} & -\frac{x}{z} \\ 0 & 0 & 0 & u & v & 1 & -u\frac{y}{z} & -v\frac{y}{z} & -\frac{y}{z} \end{bmatrix} \tag{A.110}$$

where $(u,\, v)^\top = \mathbf{u}_j^{\mathcal{C}}$ and $(x,\, y,\, z)^\top = S^{-1}\left(\mathbf{p}\right) \cdot h^{-1}\left(\mathbf{u}_j^{\mathcal{C}}\right)$.

Jacobian of Stacked Homography by State Vector

Deriving the second factor of the Jacobian,

$$\mathsf{H}_w = \frac{\partial \mathbf{p}\left(\mathbf{x}\right)}{\partial \mathbf{x}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}^{-1}}\left(\mathbf{x}\right)\right)}{\partial \mathbf{x}} \tag{A.111}$$

is more involved. To simplify, we can assume that the $\mathbf{x}$ contains only the elements involved in the computation of $\mathsf{H}^{\mathcal{C}\mathcal{A}}$, i.e., the camera pose and the feature state,

$$\mathbf{x} = \left(\mathbf{r}^{\mathcal{W}\mathcal{C}},\, \mathbf{q}^{\mathcal{W}\mathcal{C}},\, \mathbf{y}\right)^\top = \left(\mathbf{r}^{\mathcal{W}\mathcal{C}},\, \mathbf{q}^{\mathcal{W}\mathcal{C}},\, \mathbf{c},\, \boldsymbol{\phi},\, \rho,\, \boldsymbol{\theta}\right)^\top. \tag{A.112}$$

Columns of the Jacobian corresponding to other state variables are $\mathbf{0}$.

We start by deriving the Jacobian of $S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\left(\mathbf{x}\right)\right)$ before we turn to the Jacobian (A.111) of its inverse. With the abbreviated state vector introduced above, this Jacobian has the following form:

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{x}} = \left[\frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{r}^{\mathcal{W}\mathcal{C}}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{q}^{\mathcal{W}\mathcal{C}}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{c}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \boldsymbol{\phi}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \rho} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \boldsymbol{\theta}}\right]. \tag{A.113}$$

The expression for the homography is

$$\mathsf{H}^{\mathcal{C}\mathcal{A}} = \underbrace{\mathsf{K}\,\mathsf{R}_{\mathbf{q}^{\mathcal{W}\mathcal{C}}}^{-1} \underbrace{\left(\mathsf{R}_{\boldsymbol{\phi}}\mathbf{n}(\boldsymbol{\theta})^\top\mathbf{m}^{\mathcal{A}} \;+\; \rho\left(\mathbf{c} - \mathbf{r}^{\mathcal{W}\mathcal{C}}\right)\mathbf{n}(\boldsymbol{\theta})^\top\right)}_{\mathsf{B}}}_{\mathsf{H}_N}\mathsf{K}^{-1}. \tag{A.114}$$

We introduce abbreviations H_N, B for sub-terms as indicated. The Jacobians of the homography by these sub-terms are needed commonly in the parts of Equation A.113. We have

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial S\left(\mathsf{H}_N\right)} = \frac{\partial S\left(\mathsf{K}\mathsf{H}_N\mathsf{K}^{-1}\right)}{\partial S\left(\mathsf{H}_N\mathsf{K}^{-1}\right)} \cdot \frac{\partial S\left(\mathsf{H}_N\mathsf{K}^{-1}\right)}{\partial S\left(\mathsf{H}_N\right)} \tag{A.115}$$

and

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial S\left(\mathsf{B}\right)} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial S\left(\mathsf{H}_N\right)} \cdot \frac{\partial S\left(\mathsf{R}_{\mathbf{q}^{\mathcal{W}\mathcal{C}}}^{-1}\mathsf{B}\right)}{\partial S\left(\mathsf{B}\right)}. \tag{A.116}$$

The factors are applications of the "stacked matrix product" Jacobians (Equations A.97 and A.98).

For the parts of the Jacobian A.113 we obtain the following. The Jacobian with respect to the current camera translation is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{r}^{\mathcal{WC}}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \frac{\partial S\left(\rho\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial \mathbf{r}^{\mathcal{WC}}} \tag{A.117}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \rho \cdot \frac{\partial S\left(\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)} \cdot \frac{\partial\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)}{\partial \mathbf{r}^{\mathcal{WC}}} \tag{A.118}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot (-\rho) \cdot \frac{\partial S\left(\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)}, \tag{A.119}$$

where we make use of (A.116) and the Jacobian "stacked vector outer product" (Equation A.101).

The Jacobian with respect to the current camera rotation is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{q}^{\mathcal{WC}}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{H}_N\right)} \cdot \frac{\partial S\left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\mathsf{B}\right)}{\partial S\left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\right)} \cdot \frac{\partial S\left(\mathsf{R}_{\mathbf{q}^{\mathcal{WC}}}^{-1}\right)}{\partial \mathbf{q}^{\mathcal{WC}-1}} \cdot \frac{\partial \mathbf{q}^{\mathcal{WC}-1}}{\partial \mathbf{q}^{\mathcal{WC}}}. \tag{A.120}$$

where we make use of (A.115) and the Jacobians "stacked matrix product" (Equation A.97), "stacked rotation from quaternion" (Equation A.104), and "invert quaternion" (Equation A.18).

The Jacobian with respect to the feature anchor translation is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{c}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \frac{\partial S\left(\rho\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial \mathbf{r}^{\mathcal{WC}}} \tag{A.121}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \rho \cdot \frac{\partial S\left(\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)} \cdot \frac{\partial\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)}{\partial \mathbf{c}} \tag{A.122}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \rho \cdot \frac{\partial S\left(\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)} \tag{A.123}$$

$$= -\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{r}^{\mathcal{WC}}}, \tag{A.124}$$

i.e., the Jacobian with respect to $\mathbf{r}^{\mathcal{WC}}$ negated.

The Jacobian with respect to the feature anchor rotation is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \boldsymbol{\phi}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \frac{\partial S\left(\mathsf{R}_{\boldsymbol{\phi}}\mathbf{n}(\boldsymbol{\theta})^{\top}\mathbf{m}^{\mathcal{A}}\right)}{\partial S\left(\mathsf{R}_{\boldsymbol{\phi}}\right)} \cdot \frac{\partial S\left(\mathsf{R}_{\boldsymbol{\phi}}\right)}{\partial \exp\left(\boldsymbol{\phi}\right)} \cdot \frac{\partial \exp\left(\boldsymbol{\phi}\right)}{\partial \boldsymbol{\phi}} \tag{A.125}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \left(\mathbf{n}(\boldsymbol{\theta})^{\top}\mathbf{m}^{\mathcal{A}}\right) \cdot \frac{\partial S\left(\mathsf{R}_{\boldsymbol{\phi}}\right)}{\partial \exp\left(\boldsymbol{\phi}\right)} \cdot \frac{\partial \exp\left(\boldsymbol{\phi}\right)}{\partial \boldsymbol{\phi}} \tag{A.126}$$

where we make use of (A.116) and the Jacobians "stacked rotation from quaternion" (Equation A.104), "quaternion from exponential" (Equation A.34).

The Jacobian with respect to the inverse depth of the feature is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \rho} = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \frac{\partial S\left(\rho\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial \rho} \tag{A.127}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot S\left(\left(\mathbf{c}-\mathbf{r}^{\mathcal{WC}}\right)\mathbf{n}(\boldsymbol{\theta})^{\top}\right) \tag{A.128}$$

where we make use of (A.116).

Finally, the Jacobian with respect to the feature normal polar coordinates is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \boldsymbol{\theta}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \left(\frac{\partial S\left(\mathsf{R}_{\boldsymbol{\phi}} \mathbf{n}(\boldsymbol{\theta})^{\top} \mathbf{m}^{\mathcal{A}}\right)}{\partial \left(\mathbf{n}(\boldsymbol{\theta})^{\top} \mathbf{m}^{\mathcal{A}}\right)} \cdot \frac{\partial \mathbf{n}(\boldsymbol{\theta})^{\top} \mathbf{m}^{\mathcal{A}}}{\partial \mathbf{n}(\boldsymbol{\theta})} + \right.$$

$$\left. + \frac{\partial S\left(\rho \left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right) \mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial \mathbf{n}(\boldsymbol{\theta})} \right) \cdot \frac{\partial \mathbf{n}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \quad \text{(A.129)}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial S\left(\mathsf{B}\right)} \cdot \left(S\left(\mathsf{R}_{\boldsymbol{\phi}}\right) \cdot \mathbf{m}^{\mathcal{A}\top} \; + \; \rho \cdot \frac{\partial S\left(\left(\mathbf{c} - \mathbf{r}^{\mathcal{WC}}\right) \mathbf{n}(\boldsymbol{\theta})^{\top}\right)}{\partial \mathbf{n}(\boldsymbol{\theta})} \right) \frac{\partial \mathbf{n}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \quad \text{(A.130)}$$

where we make use of (A.116) and the Jacobians "stacked vector outer product" (Equation A.101), and "unit vector from azimuth-elevation" (Equation A.44).

This completes the Jacobian (A.113), i.e., the derivatives of the stacked homography $\mathsf{H}^{\mathcal{CA}}$. However, the goal is to derive the Jacobian of the stacked *inverse* homography $\mathsf{H}^{\mathcal{CA}^{-1}}$, Equation A.111.

We need the following basic identity (Harville, 1997):

Proposition A.8.1 (Derivative of an Inverse)**.** *The derivative of the inverse of a matrix* A *with respect to the scalar* x *is given by*

$$\frac{\partial \mathsf{A}^{-1}}{\partial x} = -\mathsf{A}^{-1} \frac{\partial \mathsf{A}}{\partial x} \mathsf{A}^{-1} \quad \text{(A.131)}$$

We can apply this to derive the Jacobian of $S\left(\mathsf{H}^{\mathcal{CA}^{-1}}\right)$ from the Jacobian of $S\left(\mathsf{H}^{\mathcal{CA}}\right)$ as follows. Consider $\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{x}}$ where $\mathbf{x} = (x_1, \ldots, x_n)$. This is a $9 \times n$ matrix of the partial derivatives of the 9 elements of $S\left(\mathsf{H}^{\mathcal{CA}}\right)$ by the n elements of the state vector. The i-th column contains the partial derivatives by x_i. The derivative of the matrix $\mathsf{H}^{\mathcal{CA}}$ by the scalar x_i is obtained by unstacking this column into a 3×3 matrix.

$$\frac{\partial \mathsf{H}^{\mathcal{CA}}}{\partial x_i} = S^{-1}\Big(col_i\left(\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{x}}\right)\Big) \quad \text{(A.132)}$$

Here, $col_i\left(\mathsf{M}\right)$ denotes the i-th column of matrix M.

Using Proposition A.8.1 we have

$$\frac{\partial \mathsf{H}^{\mathcal{CA}^{-1}}}{\partial x_i} = -\mathsf{H}^{\mathcal{CA}^{-1}} S^{-1}\Big(col_i\left(\frac{\partial S\left(\mathsf{H}^{\mathcal{CA}}\right)}{\partial \mathbf{x}}\right)\Big) \mathsf{H}^{\mathcal{CA}^{-1}} \quad \text{(A.133)}$$

as the derivative of $\mathsf{H}^{\mathcal{CA}^{-1}}$ by the scalar x_i.

We can apply this procedure to all $x_1 \ldots x_n$ and stack the results into the columns of the desired Jacobian

$$\mathsf{H}_w = \frac{\partial S\left(\mathsf{H}^{\mathcal{CA}^{-1}}\left(\mathbf{x}\right)\right)}{\partial \mathbf{x}} = \left[S\left(\frac{\partial \mathsf{H}^{\mathcal{CA}^{-1}}}{\partial x_1}\right) \quad \cdots \quad S\left(\frac{\partial \mathsf{H}^{\mathcal{CA}^{-1}}}{\partial x_n}\right)\right]. \quad \text{(A.134)}$$

Jacobian of Stacked Homography for the Left Eye

The difference between the measurement function for the left and right eye lies in the homography parameters which for the left eye are

$$\mathbf{p}\left(\mathbf{x}\right) = S\big(\mathsf{H}^{\mathcal{C}'\mathcal{A}^{-1}}\left(\mathbf{x}\right)\big). \tag{A.135}$$

Analogously to the right eye, we start by deriving the Jacobian of $S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\left(\mathbf{x}\right)\right)$ and use it to compute the Jacobian of its inverse afterwards. With the state vector (A.112) this Jacobian has the following form

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{x}} = \left[\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{r}^{\mathcal{W}\mathcal{C}}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{q}^{\mathcal{W}\mathcal{C}}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{c}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \boldsymbol{\phi}} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \rho} \quad \frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \boldsymbol{\theta}}\right]. \tag{A.136}$$

The homography for the left image is composed of the right image homography $\mathsf{H}^{\mathcal{C}\mathcal{A}}$ and an additive term which compensates for the shift of the left eye by the baseline $\mathbf{b} = (b, 0, 0)^\top$ (cf. Equation 6.58)

$$\mathsf{H}^{\mathcal{C}'\mathcal{A}} = \mathsf{H}^{\mathcal{C}\mathcal{A}} + \underbrace{\underbrace{\mathsf{K}\underbrace{\left(\rho \mathbf{b}\mathbf{n}(\boldsymbol{\theta})^\top\right)}_{\mathsf{H}_N^+}\mathsf{K}^{-1}}_{\mathsf{H}^+}}. \tag{A.137}$$

We introduce abbreviations H_N^+, H^+ for sub-terms as indicated. The additive term H^+ in the homography results in additive terms in the Jacobians as well. Because neither $\mathbf{r}^{\mathcal{W}\mathcal{C}}$, $\mathbf{q}^{\mathcal{W}\mathcal{C}}$, $\mathbf{y}$, $\mathbf{c}$, nor $\boldsymbol{\phi}$ appear in the additive term, we immediately have

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{r}^{\mathcal{W}\mathcal{C}}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{r}^{\mathcal{W}\mathcal{C}}} \tag{A.138}$$

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{q}^{\mathcal{W}\mathcal{C}}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{q}^{\mathcal{W}\mathcal{C}}} \tag{A.139}$$

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \mathbf{c}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \mathbf{c}} \tag{A.140}$$

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \boldsymbol{\phi}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \boldsymbol{\phi}} \tag{A.141}$$

The Jacobian with respect to the inverse depth of the feature is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \rho} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \rho} + \frac{\partial S\left(\mathsf{K}\rho \mathbf{b}\mathbf{n}(\boldsymbol{\theta})^\top \mathsf{K}^{-1}\right)}{\partial \rho} \tag{A.142}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \rho} + S\left(\mathsf{K}\mathbf{b}\mathbf{n}(\boldsymbol{\theta})^\top \mathsf{K}^{-1}\right). \tag{A.143}$$

For the remaining Jacobian with respect to the feature normal $\boldsymbol{\theta}$ we need the derivative of H^+ by H_N^+

$$\frac{\partial S\left(\mathsf{H}^+\right)}{\partial S\left(\mathsf{H}_N^+\right)} = \frac{\partial S\left(\mathsf{K}\mathsf{H}_N^+\mathsf{K}^{-1}\right)}{\partial S\left(\mathsf{H}_N^+\mathsf{K}^{-1}\right)} \cdot \frac{\partial S\left(\mathsf{H}_N^+\mathsf{K}^{-1}\right)}{\partial S\left(\mathsf{H}_N^+\right)} \tag{A.144}$$

where we make use of the "stacked matrix product" Jacobians (Equations A.97 and A.98). Then, the Jacobian with respect to the feature normal polar coordinates is

$$\frac{\partial S\left(\mathsf{H}^{\mathcal{C}'\mathcal{A}}\right)}{\partial \boldsymbol{\theta}} = \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \boldsymbol{\theta}} + \frac{\partial S\left(\mathsf{H}^{+}\right)}{\partial S\left(\mathsf{H}_N^{+}\right)} \cdot \frac{\partial S\left(\rho \mathbf{b}\mathbf{n}(\boldsymbol{\theta})^\top\right)}{\partial \boldsymbol{\theta}} \tag{A.145}$$

$$= \frac{\partial S\left(\mathsf{H}^{\mathcal{C}\mathcal{A}}\right)}{\partial \boldsymbol{\theta}} + \frac{\partial S\left(\mathsf{H}^{+}\right)}{\partial S\left(\mathsf{H}_N^{+}\right)} \cdot \rho \cdot \frac{\partial S\left(\mathbf{b}\mathbf{n}(\boldsymbol{\theta})^\top\right)}{\partial \mathbf{n}(\boldsymbol{\theta})^\top} \cdot \frac{\partial \mathbf{n}(\boldsymbol{\theta})^\top}{\partial \boldsymbol{\theta}} \tag{A.146}$$

where we make use of the Jacobians "stacked vector outer product" (Equation A.101), and "unit vector from azimuth-elevation" (Equation A.44).

Thus we have derived the Jacobian (A.136), i.e., the derivatives of the stacked homography $\mathsf{H}^{\mathcal{C}'\mathcal{A}}$. The Jacobian of the inverse homography $\mathsf{H}^{\mathcal{C}'\mathcal{A}^{-1}}$ is easily derived analogously to Equations A.132 – A.134.

This concludes the derivation of the Jacobians of the pixel measurement model for planar features.

Bibliography

Azarbayejani, A. & Pentland, A. P. (1995). Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *17*(6), 562–575.

Bailey, T. (2002). *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, Australian Centre for Field Robotics, University of Sydney.

Baker, S., Dellaert, F., & Matthews, I. (2001). Aligning images incrementally backwards. Technical Report CMU-RI-TR-01-03, Robotics Institute, Carnegie Mellon University.

Baker, S. & Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, *56*(3), 221–255.

Bayard, D. S. & Brugarolas, P. (2004). Small body GN&C research report: Estimation algorithms. Internal Document JPL D-30281, Jet Propulsion Laboratory.

Bayard, D. S. & Brugarolas, P. B. (2005). An estimation algorithm for vision-based exploration of small bodies in space. In *Proceedings of the American Control Conference (ACC)*, (pp. 4589–4595).

Bekris, K. E., Glick, M., & Kavraki, L. E. (2006). Evaluation of algorithms for bearing-only slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Bell, B. & Cathey, F. (1993). The iterated Kalman filter update as a Gauss-Newton method. *IEEE Transactions on Automatic Control*, *38*, 294–297.

Benhimane, S. & Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, volume 1, (pp. 943–948)., Sendai, Japan.

Bierman, G. J. (1977). *Factorization Methods for Discrete Sequential Estimation*, volume 128 of *Math in Science and Engineering*. New York: Academic Press.

Broida, T., Chandrashekhar, S., & Chellappa, R. (1990). Recursive 3-d motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic Systems*, *26*(4), 639–656.

Brookner, E. (1998). *Tracking and Kalman Filtering Made Easy*. Wiley-Interscience.

Buenaposada, J. M. & Baumela, L. (2002a). Real-time tracking and estimation of plane pose. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, (pp. 697–700)., Quebec, Canada. IEEE.

Buenaposada, J. M. & Baumela, L. (2002b). Speeding up ssd planar tracking by pixel selection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume I, (pp. 565–568)., Rochester, NY, USA. IEEE.

Bürgisser, P. (1996). Algebraische Komplexitätstheorie II: Schnelle Matrixmultiplikation und Kombinatorik. *Séminaire Lotharingien de Combinatoire*, *B36b*, 16pp.

Castellanos, J., Tardós, J., & Schmidt, G. (1997). Building a global map of the environment of a mobile robot: The importance of correlations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Chekhlov, D., Pupilli, M., Mayol-Cuevas, W., & Calway, A. (2007). Robust real-time visual SLAM using scale prediction and exemplar based feature description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chiuso, A., Favaro, P., Jin, H., & Soatto, S. (2002). Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *24*(4), 523–535.

Chli, M. & Davison, A. J. (2008). Active matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 5302 of *LNCS*, (pp. 72–85). Springer.

Chli, M. & Davison, A. J. (2009). Active matching for visual tracking. *Robotics and Autonomous Systems (RAS). Special Issue on Inside Data Association*.

Civera, J., Davison, A. J., & Montiel, J. M. M. (2007). Inverse depth to depth conversion for monocular SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Civera, J., Davison, A. J., & Montiel, J. M. M. (2008). Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics (T-RO)*, *24*(5), 932–945.

Clemente, L. A., Davison, A. J., Reid, I., Neira, J., & Tardos, J. D. (2007). Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems (RSS)*.

Cobzas, D., Jagersand, M., & Sturm, P. (2009). 3d ssd tracking with estimated 3d planes. *Image and Vision Computing*, *27*, 69–79.

Cobzas, D. & Sturm, P. (2005). 3d ssd tracking with estimated 3d planes. In *2nd Canadian Conference on Computer and Robot Vision (CRV)*, (pp. 129–134).

Cummins, M. & Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, *27*(6), 647–665.

Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Davison, A. J. & Murray, D. W. (1998). Mobile robot localisation using active vision. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *29*(6), 1052–1067.

Dellaert, F. & Collins, R. (1999). Fast image-based tracking by selective pixel integration. In *Workshop on Frame-Rate Vision. In conjunction with the IEEE International Conference on Computer Vision*.

Dellaert, F., Thorpe, C., & Thrun, S. (1998). Super-resolved texture tracking of planar surface patches. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

Drummond, T. & Cipolla, R. (2002). Real-time tracking of complex structures with on-line camera calibration. *Image and Vision Computing*, *20*(5-6), 427 – 433.

Eade, E. & Drummond, T. (2006a). Edge landmarks in monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Eade, E. & Drummond, T. (2006b). Scalable monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eade, E. & Drummond, T. (2007). Monocular SLAM as a graph of coalesced observations. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Eade, E. & Drummond, T. (2008). Unified loop closing and recovery for real time monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Estrada, C., Neira, J., & Tardós, J. D. (2005). Hierarchical SLAM: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics (T-RO)*, *21*(4), 588–596.

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fitzgibbon, A. W. & Zisserman, A. (1998). Automatic camera recovery for closed or open image sequences. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Forsyth, D. A. & Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference.

Fraundorfer, F., Schindler, K., & Bischof, H. (2006). Piecewise planar scene reconstruction from sparse correspondences. *Image and Vision Computing*, *24*(4), 395 – 406.

Funke, J. & Pietzsch, T. (2009a). A framework for evaluating visual SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Funke, J. & Pietzsch, T. (2009b). SLAMDUNK - visual SLAM evaluation framework.

Fusiello, A., Trucco, E., & Verri, A. (2000). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, *12*(1), 16–22.

Gee, A. P., Chekhlov, D., Calway, A., & Mayol-Cuevas, W. (2008). Discovering higher level structure in visual SLAM. *IEEE Transactions on Robotics (T-RO)*, *24*(5), 980–990.

Gee, A. P., Chekhlov, D., Mayol, W., & Calway, A. (2007). Discovering planes and collapsing the state space in visual SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Gee, A. P. & Mayol-Cuevas, W. (2006). Real-time model-based SLAM using line segments. In *International Symposium on Visual Computing (ISVC)*.

Gelb, A. (1974). *Applied Optimal Estimation*. MIT Press.

Golub, G. H. & Van Loan, C. F. (1996). *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)* (3rd ed.). The Johns Hopkins University Press.

Grassia, F. S. (1998). Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, *3*, 29–48.

Guivant, J. & Nebot, E. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation (T-RA)*, *17*(3), 242–257.

Hager, G. D. & Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *20*(10), 1025–1039.

Harris, C. & Stennet, C. (1990). RAPID - a video rate object tracker. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, (pp. 147–151).

Hartley, R. & Zisserman, A. (2000). *Multiple View Geometry in Computer Vision*. Cambridge Univerity Press.

Harville, D. A. (1997). *Matrix Algebra From a Statistician's Perspective*. Springer.

Hermann, S. & Klette, R. (2009). The naked truth about cost functions. Technical report, University of Auckland.

Irani, M. & Anandan, P. (2000). About direct methods. In B. Triggs, A. Zisserman, & R. Szeliski (Eds.), *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science* (pp. 267–277). Springer Berlin / Heidelberg.

Jin, H., Favaro, P., & Soatto, S. (2003). A semi-direct approach to structure from motion. *The Visual Computer*, *19*(6), 377–394.

Klein, G. & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

Klein, G. & Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 5303, (pp. 802–815). Springer.

Knight, J. G. H., Davison, A. J., & Reid, I. D. (2001). Towards constant time SLAM using postponement. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, volume 1, (pp. 405–413).

Konolige, K. & Agrawal, M. (2008). FrameSLAM: from bundle adjustment to realtime visual mapping. *IEEE Transactions on Robotics (T-RO)*, *24*(5), 1066–1077.

Lemaire, T., Berger, C., Jung, I.-K., & Lacroix, S. (2007). Vision-based SLAM: Stereo and monocular approaches. *International Journal of Computer Vision (IJCV)*, *74*(3), 343–364.

Lepetit, V. & Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *28*(9), 1465–1479.

Lovi, D., Birkbeck, N., Cobzas, D., & Jagersand, M. (2010). Incremental free-space carving for real-time 3d reconstruction. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, *60*, 91–110.

Lucas, B. D. & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the IEEE International Joint Conference on Artificial Intelligence (IJCAI)*, (pp. 674–679).

Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. S. (2003). *An Invitation to 3-D Vision*. Springer.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Malis, E. (2004). Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 1843–1848)., New Orleans, USA.

Martínez-Carranza, J. & Calway, A. (2009a). Appearance based extraction of planar structure in monocular SLAM. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*, (pp. 269–278).

Martínez-Carranza, J. & Calway, A. (2009b). Efficiently increasing map density in visual SLAM using planar features with adaptive measurement. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Martínez-Carranza, J. & Calway, A. (2010). Unifiying planar and point mapping in monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Matas, J., Chum, O., Urba, M., & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference (BMVC)*.

McLauchlan, P. F. & Murray, D. W. (1995). A unifying framework for structure and motion recovery from image sequences. In *Proceedings of the International Conference on Computer Vision (ICCV)*, (pp. 314–320).

Mei, C., Benhimane, S., Malis, E., & Rives, P. (2006). Constrained multiple planar template tracking for central catadioptric cameras. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Mei, C., Benhimane, S., Malis, E., & Rives, P. (2008). Efficient homography-based tracking and 3-d reconstruction for single-viewpoint sensors. *IEEE Transactions on Robotics (T-RO)*, *24*(6), 1352–1364.

Mei, C., Sibley, G., Cummins, M., Newman, P., & Reid, I. (2009). A constant time efficient stereo SLAM system. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Mei, C., Sibley, G., Cummins, M., Newman, P., & Reid, I. (2010). RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision (IJCV)*.

Molton, N. D., Davison, A. J., & Reid, I. D. (2003). Parameterisation and probability in image alignment. Technical Report OUEL 2266/2003, Department of Engineering Science, University of Oxford.

Molton, N. D., Davison, A. J., & Reid, I. D. (2004a). Locally planar patch features for real-time structure from motion. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Molton, N. D., Davison, A. J., & Reid, I. D. (2004b). Parameterisation and probability in image alignment. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*.

Montemerlo, M. & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using fastslam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the IEEE International Joint Conference on Artificial Intelligence (IJCAI)*.

Montiel, J., Civera, J., & Davison, A. J. (2006). Unified inverse depth parameterization for monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*.

Moutarlier, P. & Chatila, R. (1989). Stochastic multisensory data fusion for mobile robot location and environment modeling. In *Proceedings of the International Symposium on Robotics Research*.

Neira, J., Ribeiro, M. I., & Tardos, J. D. (1997). Mobile robot localisation and map building using monocular vision. In *Proceedings of the International Symposium on Intelligent Robotics Systems*.

Neira, J. & Tardos, J. D. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation (T-RA)*, *17*, 890–897.

Newcombe, R. A. & Davison, A. J. (2010). Live dense reconstruction with a single moving camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nickels, K. & Hutchinson, S. (2002). Estimating uncertainty in ssd-based feature tracking. *Image and Vision Computing*, *20*, 47–58.

Nistér, D., Naroditsky, O., & Bergen, J. (2006). Visual odometry for ground vehicle applications. *Journal of Field Robotics*, *23*, 3–20.

Pan, Q., Reitmayr, G., & Drummond, T. (2009). ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Paz, L., Piniés, P., Tardós, J., & Neira, J. (2008). Large scale 6DOF SLAM with a stereo camera in hand. *IEEE Transactions on Robotics (T-RO)*, *24*(5), 946–957.

Paz, L. M. (2008). *EKF SLAM in O(n)*. PhD thesis, Universidad de Zaragoza.

Persistence of Vision Pty. Ltd. (2004). Persistence of Vision (TM) Raytracer.

Pietzsch, T. (2008a). Efficient feature parameterisation for visual SLAM using inverse depth bundles. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Pietzsch, T. (2008b). Planar Features for Visual SLAM. In *Proceedings of the German Conference on Artificial Intelligence (KI)*. Springer.

Piniés, P., Paz, L. M., & Tardós, J. (2009). CI-Graph: an efficient approach for large scale SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Piniés, P. & Tardós, J. D. (2008). Large scale slam building conditionally independent local maps: Application to monocular vision. *IEEE Transactions on Robotics (T-RO)*, *24*(5), 1094–1106.

Pollefeys, M., Nister, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewenius, H., Yang, R., Welch, G., & Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision (IJCV)*, *78*(2), 143–167.

Pupilli, M. & Calway, A. (2006). Real-time visual SLAM with resilience to erratic motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rosten, E. & Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 1, (pp. 430–443).

Roweis, S. & Ghahramani, Z. (1999). A unifying review of linear gaussian models. *Neural Computation*, *11*(2), 305–345.

Scharstein, D. & Szeliski, R. (2009). Middlebury computer vision pages.

Se, S., Lowe, D., & Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, *21*(8), 735–758.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 519–528).

Shi, J. & Tomasi, C. (1994). Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 593–600).

Sibley, G., Mei, C., Reid, I., & Newman, P. (2009). Adaptive relative bundle adjustment. In *Proceedings of Robotics: Science and Systems (RSS)*.

Silveira, G., Malis, E., & Rives, P. (2008). An efficient direct approach to visual slam. *IEEE Transactions on Robotics (T-RO)*, *24*(5), 969 –979.

Silveira, G. F., Malis, E., & Rives, P. (2006). Real-time robust detection of planar regions in a pair of images. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, (pp. 49–54).

Smith, P., Reid, I., & Davison, A. (2006). Real-time monocular SLAM with straight lines. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Smith, R., Self, M., & Cheeseman, P. (1986). Estimating uncertain spatial relationships in robotics. In *Proceedings of the Second Annual Conference on Uncertainty in Artificial Intelligence*, (pp. 435–461).

Smith, R., Self, M., & Cheeseman, P. (1988). A stochastic map for uncertain spatial relationships. In *Proceedings of the International Symposium on Robotics Research*, (pp. 467–474).

Solà, J., Monin, A., & Devy, M. (2007). BiCamSLAM: Two times mono is more than stereo. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 4795–4800). IEEE.

Solà, J., Monin, A., Devy, M., & Lemaire, T. (2005). Undelayed initialisation in bearing only slam. In *IROS*.

Strasdat, H., Montiel, J., & Davison, A. J. (2010). Scale drift-aware large scale monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*.

Strassen, V. (1969). Gaussian elimination is not optimal. *Numerische Mathematik*, *14* (3), 354–356.

Stühmer, J., Gumhold, S., & Cremers, D. (2010). Real-time dense geometry from a handheld camera. In *Proceedings of the Symposium of the German Association for Pattern Recognition (DAGM)*.

Tardós, J. D., Neira, J., Newman, P. M., & Leonard, J. J. (2002). Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, *21*, 311–330.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Thrun, S., Liu, Y., Koller, D., Ng, A., Ghahramani, Z., & Durrant-Whyte, H. (2004). Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, *23*, 693–716.

Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (1999). Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms. In conjunction with the IEEE International Conference on Computer Vision*, (pp. 298–372). Springer.

Tully, S., Moon, H., Kantor, G., & Choset, H. (2008). Iterated filters for bearing-only SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (pp. 1442–1448).

Weingarten, J. & Siegwart, R. (2006). 3d SLAM using planar segments. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., & Tardos, J. (2008). An image-to-map loop closing method for monocular SLAM. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, (pp. 2053–2059).

Williams, B., Klein, G., & Reid, I. (2007). Real-time SLAM relocalisation. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Williams, S. B., Dissanayake, G., & Durrant-Whyte, H. (2002). An efficient approach to the simultaneous localisation and mapping problem. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Wuest, H., Pagani, A., & Stricker, D. (2007). Feature management for efficient camera tracking. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, (pp. 769–778)., Berlin, Heidelberg. Springer.

Wuest, H., Wientapper, F., & Stricker, D. (2008). Acquisition of high quality planar patch features. In *Proceedings of the 4th International Symposium on Advances in Visual Computing*, (pp. 530–539)., Berlin, Heidelberg. Springer-Verlag.